OLD DOMINION UNIVERSITY RESEARCH FOUNDATION

DEPARTMENT OF MATHEMATICAL AND COMPUTING SCIENCES
SCHOOL OF SCIENCES
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA

Technical Report

FINITE ELEMENT COMPUTATION OF A VISCOUS COMPRESSIBLE FREE SHEAR FLOW GOVERNED BY THE TIME DEPENDENT NAVIER-STOKES EQUATIONS

By

Charlie H. Cooke

and

Doris K. Blanchard

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia

Under
Grant NSG 1098
September 9, 1974 - September 9, 1975

October 1975

FINITE ELEMENT COMPUTATION OF A VISCOUS COMPRESSIBLE FREE SHEAR FLOW GOVERNED BY THE TIME DEPENDENT NAVIER-STOKES EQUATIONS

By

Charlie H. Cooke[1] and Doris K. Blanchard[2]

SUMMARY

A finite element algorithm for solution of fluid flow problems characterized by the two-dimensional compressible Navier-Stokes equations has been developed. The program is intended for viscous compressible high speed flows; hence, primitive variables are utilized. The physical solution is approximated by trial functions which at a fixed time are piece-wise cubic on triangular elements. The Galerkin technique is employed to determine the finite-element model equations. A leapfrog time integration is used for marching asymptotically from initial to steady state, with iterated integrals evaluated by numerical quadratures. The nonsymmetric linear systems of equations governing time transition from step-to-step are solved using a rather economical block iterative triangular decomposition scheme.

Proof of concept has been accomplished by the numerical computation of a free shear flow. Numerical results of the finite-element method are in excellent agreement with those obtained from a finite difference solution of the same test problem.

---

[1] Associate Professor of Mathematical and Computing Sciences, Old Dominion University, Norfolk, VA 23508.

[2] High-Speed Aerodynamics Division, NASA Langley Research Center, Hampton, Virginia 23665.

INTRODUCTION

Over the past two decades the finite element method has become a widely accepted tool for obtaining reliable numerical solutions to problems in structural mechanics. However, in fluids calculations the method can at best be regarded as relatively untested, although its application is steadily diversifying. Recent endeavors include analysis of flow problems involving free surfaces, such as occur in groundwater seepage and oil depletion problems in the petroleum industry, which involve the nonlinear gas flow equations (ref. 1); wind driven lake circulation problems including islands (ref. 2); limited region modeling of mesoscale phenomena associated with the dynamics of ocean circulation (ref. 3); calculation of flows over nonlifting circular airfoils, based on the small disturbance nonlinear transonic flow equations of an inviscid compressible fluid (ref. 4); laminar three-dimensional boundary layer flow of a multicomponent compressible fluid (ref. 5); and pressure distributions for confined flow problems (ref. 6).

For the most part these investigations have considered low Reynolds number low speed flows. Sparseness in problem dimension and number of dependent variables (in effect, no more than two space dimensions and/or two dependent variables) has been a simplifying characteristic. The governing equations permitted symmetric equation solvers when the problems were implicit. The present investigation, to the best of the authors' knowledge, is one of the first attempts to solve by finite elements a fluid dynamics problem characterized simultaneously by a variety of cumbersome aspects tending to severely complicate the numerical formulation. The flow is governed by the time-dependent nonlinear non-self adjoint compressible Naiver-Stokes equations, in two space dimensions and three dependent variables. (The mitigating assumption of constant total temperature, i.e., adiabatic jet mixing, permits the energy equation to be replaced by an algebraic equation.)

A sophisticated higher order element, cubic B-splines on triangles, is employed. The nature of the problem forces area integrals arising from the Galerkin formulation to be evaluated by numerical quadratures. The resulting systems of finite element equations are nonsymmetric, and the problem size dictated development of a linear system solver specially adapted to the characteristics of the problem.

The goal of the present investigation is two-fold; first, the development of a reliable numerical tool for computing laminar flows which can be extended for use in testing fully two-dimensional turbulence models for a wide range of free shear flow applications, such as interference heating, separated flows, jet exhaust noise reduction, combustor design, and tangential slot injection. Second, an assessment of the feasibility of using the finite element method as a tool for fluid mechanic calculations is an expected product of the research. The same problem has been solved by various finite difference schemes, and computational results are readily available (ref. 7) for comparison.

In terms of computer core size requirements, as well as efficiency per computational step, the explicit finite difference methods afford the most economical approach to solving the time-dependent viscous compressible Navier-Stokes equations. However, for numerical stability very small time steps relative to spatial grid size are required. This mandates long computation time to reach steady-state, especially for flows characterized by fine mesh spacing in regions of large gradients. As a result, the unconditionally stable implicit alternating-direction methods (ADI), Hopscotch ADE methods, and MacCormack's method (ref. 8) have been found currently the most popular alternatives. Although the practical advantage of ADI methods over explicit methods is nowhere near that predicted by the Von Neuman analysis, experience indicates large time steps are allowed, although somewhat less than an order of magnitude larger than the explicit CFL limit.

The features of the finite element method which appear to enhance its attractiveness for fluid dynamics applications, and which are distinct advantages over other numerical schemes, include:

1. Complex boundaries and boundary conditions can be easily and effectively treated. For example, by using higher order elements which carry function and derivative values as unknowns, both function and derivative boundary conditions can be treated homogeneously, without extra boundary error generated by writing finite difference derivative approximations. Also, triangular elements allow more precise treatment of complex geometry, and even further precision is obtained through isoparametric elements.

2. Mesh refinement in regions of large gradients is easily implemented.

3. The approximations for a particular problem are more flexible. These are reflected in freedom of choice over element shape, size, and order of approximation in each element.

4. Time steps at least as large as allowed by ADI are permitted.

In general, it can be said that the approach affords a fairly automatic scheme which does not leave much flexibility for manipulation of the mechanics of the algorithm, i.e., the kind of differencing for individual terms is not readily externally visible.

On the other hand, it is clear that the advantages of the method, flexibility and reliability, come with a price attached. For example, the complexity of implementation of the method entails much more sophistication of computer code, longer development time, more manhour and machine hour expense, etc. Aside from development of the fundamental numerical algorithm, myriad data manipulation and supporting software programs need be produced for effective use of the method. Moreover, the execution time per computational step is relatively expensive.

FLUID DYNAMICS MODEL OF A FREE SHEAR FLOW

A fluid dynamics problem whose description requires the complete Navier-Stokes equations is free mixing flow with no dominant flow direction. Such a problem occurs in the mixing of a supersonic jet with an imposed crossflow. This problem would embody some of the complications often unavoidable in practical calculation of flows for real vehicles, such as sharp corners, truncated computational domain, computational boundary conditions at artificial downstream boundaries, and boundary conditions for mixed parabolic-hyperbolic flow. Such a problem has been attempted by finite difference methods (ref. 7), and it is intended to apply the finite element program for future comparisons of the solutions to this problem.

However, since the individual effects of such complications are hard to isolate, the pioneer application of the finite element program developed under NASA contract NAS1-11707-37 is to the computation of a free shear flow generated by the parallel mixing of two supersonic jets, initially separated by a thin splitter plate. Solution of such a problem does not in this case require the full Navier-Stokes equations, since fairly accurate results can be obtained using the quasi-parallel assumptions of parabolic boundary layer theory (ref. 9). However, the availability of solutions generated by several computational methods affords a ready basis for evaluation of the finite element method.

Flow Field Configuration

The flow field configuration of the problem considered is shown in figure 1. The computational domain begins downstream from the base of the splitter plates. For the test case presented in the present paper the jet Mach numbers are 3 and 1.68 for Re = 1000.

Governing Equations

Steady-state flow is asymptotically approached through solution of the time-dependent Navier-Stokes equations. The assumption of constant total temperature (adiabatic mixing) and two-dimensional flow yields the following non-dimensional systems of governing equations:

Continuity:

$$\frac{\partial \rho}{\partial t} + \rho \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) + v \frac{\partial \rho}{\partial y} + u \frac{\partial \rho}{\partial x} = 0 \tag{1}$$

y-momentum:

$$-\rho \left(\frac{\partial v}{\partial t} + v \frac{\partial v}{\partial y} + u \frac{\partial v}{\partial x}\right) - \frac{\partial P}{\partial y} + \frac{4}{3R_s} \frac{\partial}{\partial y} \left(\mu \frac{\partial v}{\partial y}\right) - \frac{\partial}{\partial y} \left(\frac{2\mu}{3R_s} \frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial x} \left[\frac{\mu}{R_s} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] = 0 \tag{2}$$

x-momentum:

$$-\rho \left(\frac{\partial u}{\partial t} + v \frac{\partial u}{\partial y} + u \frac{\partial u}{\partial x}\right) - \frac{\partial P}{\partial x} + \frac{4}{3R_s} \frac{\partial}{\partial x} \left(\mu \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial x} \left(\frac{2\mu}{3R_s} \frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial y} \left[\mu \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right] = 0 \tag{3}$$

Temperature relation:

$$T = 1 - u^2 - v^2 \tag{4}$$

Constitutive relationships:

Sutherland's viscosity law: $$\mu = T^{3/2}\left[\frac{T_s + 198.6}{T_s T + 198.6}\right] \tag{5}$$

Perfect gas law: $$P = \rho\left(\frac{\gamma - 1}{2\gamma}\right) T \;. \tag{6}$$

In equation (5), μ, T are dimensionless, although T_s and the constant 198.6 (Sutherland's constant) are expressed in degrees Rankine. The variables used to non-dimensionalize eqs. (1) to (6) are presented in reference 7.

Throughout this paper, the notation f_x and f_y denote quantities associated with the x,y directions; the convention for first partial derivatives will be

$$\frac{\partial f}{\partial x} = f_{,x} \qquad \text{or} \qquad \frac{\partial f}{\partial y} = f_{,y} \;. \tag{7}$$

Boundary Conditions

Boundary conditions for the problem are shown schematically in figure 2. Function specifications are given for all three variables on the inflow; symmetry conditions apply at the bottom, and on the top function specification is made for velocity, zero normal derivative for density. On the outflow a computational boundary condition is applied; this could be either linear or quadratic extrapolation. The downstream boundary condition computations will be discussed in detail in a subsequent section of the present paper.

FINITE ELEMENT APPROXIMATION

Our approximation to the fluid dynamics problem [equations (1) to (7) and boundary conditions of figure 2] is obtained by applying the classical Galerkin (or method of weighted residuals) in conjunction with finite elements. The first step is to triangulate the computational domain Ω with boundary Γ, and then consider piecewise polynomial approximating (trial) functions on this grid.

Trial Functions

For purposes of illustration, the trial functions for approximating density variations are of the form

$$\rho(x,y,t) = \sum_{j=1}^{N} \rho_J(t)\ \phi_J(x,y) \quad . \tag{8}$$

Here N is the total number of nodes. The (B-spline) functions $\{\phi_J\}$ comprise a local and interpolating basis for functions of the form equation (8), which are continuous and piecewise cubic polynomial on Ω, with sectionally continuous first partial derivatives, which are infinitely differentiable cubic on the interior of each triangle. (For a more precise description of the B-splines, see reference 10).

The weights $\rho_J(t)$ are chosen by Galerkin's methods. Thus, the final approximating function satisfies all boundary and initial conditions at problem nodes, and approximately satisfies the governing equations over the domain. For each trial function (density and two velocity components) there are ten nodes per triangle; triple nodes at triangle vertices, and a single node at the centroid (see fig. 3). The parameters ρ_J each associate with a distinct node, and represent approximations to function and first partial derivative values $(\rho, \rho_{,x}, \rho_{,y})$ at vertices, and function values alone at the centroid.

Time Discretization

For simplicity, the time discretization will be indicated only for the pseudo-viscous continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \, \overline{U} = \varepsilon \, \nabla^2 \rho \quad , \tag{9}$$

where $\overline{U}$ is the velocity vector. (For ε non-zero but small, artificial viscosity is added to the continuity equation.)

The weak form of (9) is obtained upon multiplying by an arbitrary function and integrating over Ω. Applying Green's theorem to remove the second derivative from the equation one obtains

$$\iint_\Omega \left\{ \phi_J \frac{\partial \rho}{\partial t} + (\overline{U} + \varepsilon \nabla \phi_J) \cdot \nabla \rho \right\} dA = \varepsilon \oint_\Gamma \phi_J \frac{d\rho}{d\overline{N}} \, d\ell \quad , \tag{10}$$

where $\frac{d\rho}{d\overline{N}}$ is the normal derivative on Γ (positive in the outward direction).

The time discretization chosen employs central differences on time derivatives and time averages of space gradients in ρ. Denoting the time step by τ and the time index by $n (t_n = t_o + n\tau)$,

$$\begin{aligned} &\iint_\Omega \left\{ \phi_J \frac{[\rho_{n+1} - \rho_{n-1}]}{\tau} + (\overline{U}_n + \varepsilon \nabla \phi_J) \cdot (\nabla \rho_{n+1} + \nabla \rho_{n-1}) \right\} dA \\ &= \varepsilon \oint_\Gamma \phi_J \left[\left(\frac{d\rho}{d\overline{N}} \right)_{n+1} + \left(\frac{d\rho}{d\overline{N}} \right)_{n-1} \right] d\ell + 0(\tau^2) \end{aligned} \tag{11}$$

If ϕ_J were the B-spline associated with a node at which ρ_J is unknown (no boundary condition applies), substitution of equation (8) into (11) would yield the finite element equation corresponding to the density variable at this node. The velocity equations are obtained by a similar procedure. The chosen time discretization has the advantage of yielding (implicit) linear systems of finite element equations, second order in time and fourth order in space, except at the downstream boundary, where the second order accurate quadratic (or first order accurate linear) extrapolation boundary condition applies.

Local Equations

On a given triangle Ω_e the trial function for density may be written as

$$\rho_e(x,y,t) = \sum_{\ell=1}^{10} \rho_\ell \, \phi_\ell = \bar{\rho}^T \, \Phi \quad , \tag{12}$$

where the ρ_ℓ are density parameters associated with nodes on this element, and ϕ_ℓ are the corresponding B-splines, with

$$\begin{aligned} \bar{\rho}^T &= (\rho_1, \rho_2, \ldots, \rho_{10}) \\ \Phi^T &= (\phi_1, \phi_2, \ldots, \phi_{10}) \ . \end{aligned} \tag{13}$$

With similar definitions for velocity components,

$$\begin{aligned} u &= \sum_{\ell=1}^{10} u_\ell \phi_\ell = \bar{u}^T \, \Phi \\ v &= \sum_{\ell=1}^{10} v_\ell \phi_\ell = \bar{v}^T \, \Phi \quad , \end{aligned} \tag{14}$$

the time discretized finite element equation contributions from the triangle Ω_e are as indicated below. A more detailed derivation of these equations is presented in reference 10.

Continuity equation - Contribution from a single element (No artificial viscosity)

$$\left[\iint_{\Omega_e} \phi_J \Phi^T \, dxdy - \tau \left(\bar{v}_n^T \iint_{\Omega_e} \phi_{J,y} \Phi\Phi^T \, dxdy + \bar{u}_n^T \iint_{\Omega_e} \phi_{J,x} \Phi\Phi^T \, dxdy\right)\right.$$

$$\left. - \tau\left(\bar{v}_n^T \int_{\Gamma_e} \phi_J \Phi\Phi^T \, dx - \bar{u}_n^T \int_{\Gamma_e} \phi_J \Phi\Phi^T \, dy\right)\right] \bar{\rho}_{n+1} = \left[\iint_{\Omega_e} \phi_J \Phi^T \, dxdy\right.$$

$$+ \tau\left(\bar{v}_n^T \iint_{\Omega_e} \phi_{J,y} \Phi\Phi^T \, dxdy + \bar{u}_n^T \iint_{\Omega_e} \phi_{J,x} \Phi\Phi^T \, dxdy\right) \tag{15}$$

$$\left. + \tau\left(\bar{v}_n^T \int_{\Gamma_e} \phi_J \Phi\Phi^T \, dx - \bar{u}_n^T \int_{\Gamma_e} \phi_J \Phi\Phi^T \, dy\right)\right] \bar{\rho}_{n-1}$$

x-momentum equation - Single element contribution

$$\bar{\rho}_n^T \left\{\left(\iint_{\Omega_e} \phi_J \Phi\Phi^T \, dxdy\right) \bar{u}^{n+1} + \tau \left[\sum_{\ell=1}^{10} v_\ell^n \iint_{\Omega_e} \phi_J \phi_\ell \Phi\Phi^T_{,y} \, dxdy\right] \bar{u}^{n+1}\right.$$

$$\left. + \tau\left[\sum_{\ell=1}^{10} u_\ell^n \iint_{\Omega_e} \phi_J \phi_\ell \Phi\Phi^T_{,x} \, dxdy\right] \bar{u}^{n+1}\right\} \tag{16a}$$

$$+ \frac{\tau}{R_s}\left[\left(\iint_{\Omega_e} \phi_{J,y} \mu^n \Phi^T_{,y} \, dxdy\right)\bar{u}^{n+1} + \left(\iint_{\Omega_e} \phi_{J,y} \mu^n \Phi^T_{,x} \, dxdy\right)\bar{v}^{n+1}\right]$$

$$+ \frac{2\tau}{3R_s}\left[\left(2\iint_{\Omega_e} \mu^n \phi_{J,x} \Phi^T_{,x} \, dxdy\right)\bar{u}^{n+1} - \left(\iint_{\Omega_e} \mu^n \phi_{J,x} \Phi^T_{,y} \, dxdy\right)\bar{v}^{n+1}\right]$$

(cont'd.)

$$+ \frac{\tau}{R_s}\left[\left(\oint_{\Gamma_e} \phi_J \mu^n \Phi^T_{,y}\, dx\right)\bar{u}^{n+1} + \left(\oint_{\Gamma_e} \phi_J \mu^n \Phi^T_{,x}\, dx\right)\bar{v}^{n+1}\right]$$

$$- \frac{2\tau}{3R_s}\left[2\left(\oint_{\Gamma_e} \mu^n \phi_J \Phi^T_{,x}\, dy\right)\bar{u}^{n+1} - \left(\oint_{\Gamma_e} \mu^n \phi_J \Phi^T_{,y}\, dy\right)\bar{v}^{n+1}\right]$$

$$= \bar{\rho}^T_n \left\{\left(\iint_{\Omega_e} \phi_J \Phi\Phi^T\, dxdy\right)\bar{u}^{n-1} - \tau\left[\sum_{\ell=1}^{10} v^n_\ell \iint_{\Omega_e} \phi_J \phi_\ell \Phi\Phi^T_{,y}\, dxdy\right]\bar{u}^{n-1}\right.$$

$$\left. - \tau\left[\sum_{\ell=1}^{10} u^n_\ell \iint_{\Omega_e} \phi_J \phi_\ell \Phi\Phi^T_{,x}\, dxdy\right]\bar{u}^{n-1}\right\} + 2\tau \iint_{\Omega_e} \underline{P}^n\, \phi_{J,x}\, dxdy$$

$$- \frac{\tau}{R_s}\left[\left(\iint_{\Omega_e} \phi_{J,y} \mu^n \Phi^T_{,y}\, dxdy\right)\bar{u}^{n-1} + \left(\iint_{\Omega_e} \phi_{J,y}\, \mu^n \Phi^T_{,x}\, dxdy\right)\bar{v}^{n-1}\right]$$

$$- \frac{2\tau}{3R_s}\left[\left(2\iint_{\Omega_e} \mu^n \phi_{J,x} \Phi^T_{,x}\, dxdy\right)\bar{u}^{n-1} - \left(\iint_{\Omega_e} \mu^n \phi_{J,x} \Phi^T_{,y}\, dxdy\right)\bar{v}^{n-1}\right]$$

(16a)
(concl'd.)

$$- \frac{\tau}{R_s}\left[\left(\oint_{\Gamma_e} \phi_J \mu^n \Phi^T_{,y}\, dx\right)\bar{u}^{n-1} + \left(\oint_{\Gamma_e} \phi_J \mu^n \Phi^T_{,x}\, dx\right)\bar{v}^{n-1}\right]$$

$$+ \frac{2\tau}{3R_s}\left[2\left(\oint_{\Gamma_e} \mu^n \phi_J \Phi^T_{,x}\, dy\right)\bar{u}^{n-1} - \left(\oint_{\Gamma_e} \phi_J \mu^n \Phi^T_{,y}\, dy\right)\bar{u}^{n-1}\right]$$

$$- 2\tau \oint_{\Gamma_e} P^n \phi_J\, dy$$

<u>y-momentum (16b)</u>

The contribution to the y-momentum equations from a single element may be obtained from the x-momentum equations by interchanging x and y, then reversing the algebraic signs of the boundary integral terms. These equations are asymmetric in x and y, when written with the boundary terms changed to area integrations using Green's theorem for the plane.

Boundary integrals vanish when ϕ_J is not associated with a boundary node. When ϕ_J is so associated, all boundary integrals over portions of element boundaries which are not coincident with region boundaries cancel in pairs.

Numerical Quadratures

The iterated integrals of equations (15) to (16) are in all cases inconvenient, and in the case of nonlinearities impossible, to integrate without numerical quadratures. The question of what quadrature scheme to use, and what order of accuracy it must possess, now arises.

If the order of accuracy is insufficient, the overall accuracy of the method is lower than otherwise obtainable with cubic elements. Hence one very redeeming feature of the cubic element is vitiated; the same accuracy could be achieved more economically from use of lower order elements whose accuracy matches that of the quadrature scheme.

The question of proper order of the quadrature-scheme for non-degradation of the built-in accuracy of the finite element algorithm has been investigated by Fix (ref. 11). For the present case, the quadrature scheme should be exact for two-dimensional polynomials of total degree six.

One quadrature scheme which meets this requirement is a 16-point scheme of the form

$$\iint_{\Delta} f(\xi,n)\,d\xi dn = \sum_{J=1}^{4} \left\{ W_{J1} \left[f(\xi_j, y_1\xi_j) + f(\xi_j, -y_1\xi_j) \right] + \omega_{J2} \left[f(\xi_j, y_2\xi_j) + f(\xi_j, -y_2\xi_J) \right] \right\} \tag{17}$$

where the quadrature points and weights W_{Ji} are found in table IV, page 134 of reference 12. However, the accuracy requirement is overfulfilled, and the efficiency of the algorithm suffers in this aspect.

Here the integration is over a standard triangle, with vertices of (0,0), (1,-1), and (1,1). Integrals in (x,y) space are obtained from the above upon multiplication by the magnitude of the Jacobian determinant of the appropriate affine transformation which maps an element onto the standard triangle. When derivatives appear in (x,y) space integrands, the corresponding (ξ,η) space terms is obtained by the chain rule.

For line integrals, transformation to the standard triangle yields an integral

$$\int_{-1}^{1} f(1,\eta)\,d\eta \tag{18}$$

which can be evaluated sixth order accurate with a third order Gaussian quadrature scheme.

Global Equations

The linear systems describing time transition of the numerical solution, which result from global assembly of equations (15) to (16), are

Continuity

$$D\bar{\bar{\rho}}_{n+1} = \bar{F}_{n,n-1} \tag{19}$$

Momentum

$$\left[\begin{array}{c|c} zz & zR \\ \hline Rz & RR \end{array}\right]_n \left[\begin{array}{c} \bar{\bar{u}} \\ \hline \bar{\bar{v}} \end{array}\right]_{n+1} = \left[\begin{array}{c} \bar{G} \\ \hline \bar{H} \end{array}\right]_{n,n-1} \tag{20}$$

The vectors $\bar{\bar{\rho}}$, $\bar{\bar{u}}$, $\bar{\bar{v}}$ contain all unknown nodal density and velocity variables; indices indicate time dependency of the corresponding computations.

The matrices D, zz, zR, Rz, RR are nonsymmetric and variable banded, with symmetric profile. Velocity associated matrices each have identical profiles prior to allowance for any matrix fill on LU decomposition. After such allowance zz, RR and zR, Rz are respectively pairwise identical in profile. Storage and handling of these matrices is discussed in Appendix B, under mesh generation and mesh associated data arrays.

Matrices employed in the solution process are triangularized by an LU decomposition routine, especially coded to account for lack of symmetry and matrix profile. The continuity equations are solved by the usual LU techniques. However, a special equation solver, the block iterative LU solver, was developed for momentum solution (see Appendix A). This hybrid method employs the previously described LU routine, along with an iterative procedure, to accomplish the momentum solution with decomposition only of zz, RR.

Computational Boundary Conditions

For flow problems in which the extent of the computational domain is dictated by storage restrictions of the computer, closing the problem often involves the enforcement of an artificial boundary condition on a truncated problem domain. The forced vanishing of a higher order derivative, although seemingly unnatural, is a condition often applied in computational fluid dynamics (ref. 8).

For example, the motivation behind linear extrapolation as a downstream continuation is the idea that sufficiently far downstream the flow variables should vary linearly with outflow direction. This would imply a vanishing second derivative in the vicinity of the outflow boundary. Should one force instead the vanishing of third derivatives, the result is quadratic extrapolation.

Consider a finite element triangle oriented with one edge aligned with the outflow boundary and one edge perpendicular to it, with vertices at points $P_1(x_n, y_m)$, $P_2(x_{n+1}, y_m)$, $P_3(x_{n+1}, y_{m+1})$ with $x_{n+1} = x_n + h$. A cubic trial function on this triangle has the form

$$f(x,y) = A + By + Cy^2 + Dy^3 + (E + Fy + Gy^2)x + (H + Iy)x^2 + Jx^3 \quad . \tag{21}$$

The condition (trapezoidal rule)

$$f_{n+1} = f_n + \frac{h}{2}\left[\left(\frac{\partial f}{\partial x}\right)_n + \left(\frac{\partial f}{\partial x}\right)_{n+1}\right] \tag{22}$$

forces J to vanish; it represents the finite element counterpart of quadratic extrapolation, as applied in finite difference settings.

The counterpart of linear extrapolation is not so easily achieved. For example, the extra condition

$$\left.\frac{\partial f}{\partial x}\right)_{n+1} = \left.\frac{\partial f}{\partial x}\right)_n \tag{23}$$

yields the (Euler integration) formula

$$f_{\underline{n+1}} = f_{\underline{n}} + h\left(\frac{\partial f}{\partial x}\right)_n \quad . \tag{24}$$

Conditions (22) and (23) force J, together with $H + Iy_m$, to vanish. This produces linearity over the triangle base, but <u>not</u> at other points.

True linear extrapolation would require $I = 0$ as well; a condition not naturally enforceable. Linearity of the trial functions near the outflow could only be achieved by employing a patch basis (the replacement of cubic functions by linear basis functions in this region), which would somewhat complicate the program. In the absence of this extreme, the coefficient $I(y - y_m)$ of (21) can be kept small, and linearity approximately achieved, by using a fine mesh grading in the y-direction.

It is concluded that quadratic extrapolation is the more natural of the two. Further support for this conclusion is provided by a Taylor's series analysis; even were true linearity achievable, the quadratic extrapolation affords an extra order of accuracy. This is particularly appealing, inasmuch as inaccuracy of the extrapolation can build a numerical boundary layer of error.

Furthermore, from a physical point of view, near-boundary distortion to some degree is to be expected from linear extrapolation, due to enforced outflow derivative equality. While linearity may in some cases be a physically well-grounded assumption, one would not expect it to hold in a shear layer, even in steady state. Thus, non-normal diffusive forces are to be expected, since derivatives are unnaturally modified.

Finally, the most compelling argument in favor of quadratic extrapolation is the following: As clearly seen from equations (22) to (24), were linear physics present, the quadratic extrapolation does not preclude its being modelled; whereas, in the opposite circumstance, the linear extrapolation can indeed force unnatural distortion, the degree of which will depend upon near boundary mesh refinement.

Comparison of Results for FEM and ADI Computations of a Free Shear Flow

The finite element code initially developed under contract NAS1-11707-37 was first applied to the jet mixing problem described in reference 7. However, program structure prohibited mesh refinement extensive enough to resolve the flow, resulting from in-core storage of all system matrices. As a consequence, the code has been reconstructed to provide greater mesh refinement capability. Solution of the continuity equations is completed prior to commencing assembly of the momentum equations, which frees space occupied by the D-matrix. During momentum assembly, further core is made available by building up the zR and Rz matrices on a disk storage device. The inconsummate amount of manual labor associated with providing the finite element triangles and mesh associated data arrays is now replaced by the automatic mesh generation package discussed in Appendix B.

Due to suspicions concerning the physically well-posed nature of the "wall-jet" problem, by decision of the technical monitor the first application of the restructured code has been fluid dynamic calculations associated with a parallel two stream mixing (see figs. 1 and 2). In this section numerical results are presented, as well as comparisons with the ADI solution for the same physical problem (ref. 7). The major difference in problem formulation between the two methods concerns computational boundary conditions; quadratic extrapolation for the finite element method (FEM), as opposed to linear extrapolation for ADI.

Differences in Domain

The ADI method was applied on a 10 x 122 grid point uniform rectangular mesh, with unit increment $h = .025$. For FEM purposes, economy was achieved by truncating at the top, leaving a 10 x 100 grid point mesh. However, the final FEM mesh was much coarser.

The 10 x 100 mesh was triangulated as follows: First, a coarse rectangular grid was superimposed; then each rectangle was dissected to form two triangles. In the x-direction the grid was rather coarse; increments were respectively of lengths 3h, 4h, 2h. In the y-direction, mesh increments were 3(3h) spacings; 22(2h) spacings; 1(3h) followed by 9(4h) spacings, ending with 1(3h) and 2(2h) spacings. The result is a non-uniform mesh, more refined in the shear layer; near the outflow (to afford more accuracy for the quadratic extrapolation); and at the top.

This triangulation produced a 4 x 39 rectangular mesh yielding 228 triangles, characterized by a maximum of 672 possible unknowns per dependent variable (see fig. 3 for placement of FEM unknowns on a triangle). Actually, application of boundary conditions reduces the maximum number of unknowns per dependent variable to 612. In contrast, for this same domain and the uniform mesh, ADI involves approximately 900 unknowns per dependent variable.

Boundary Condition Differences

Along the top, steady state converged ADI function values were employed as boundary values for FEM calculations. Except for the type of downstream continuation, other boundary treatment and function boundary values were identical. Derivative boundary values were obtained from a spline fitting routine.

For initial flow field, interior values were obtained by linear interpolation, with function values of the outflow differing by as much as 50 percent in function (and much larger in derivative) values from the expected steady-state results.

Artificial Viscosity

An artificial viscosity term characterized by $\varepsilon = .0001$ [see eq. (9)] was added to the continuity equation. Since a stability analysis (ref. 10) indicates marginal stability for the FEM formulation of the continuity equation, this theoretically advisable but practically negligible safety factor was deemed necessary. An order of magnitude larger ε is required to produce noticeable changes, and two orders of magnitude larger to produce maximum one percent changes, in ten steps; hence, for practical purposes it would appear the artificial viscosity can be ignored (none was used in the corresponding ADI calculations).

Numerical Results

Steady state results, for computations with $Re = 1000$, $\varepsilon = .0001$, and quadratic continuation, were essentially achieved with the following sequence of time step sizes 14(.001), 7(.002), 35(.005), 49(.01), for a total of 108 iterations. The program was allowed to run further, and numerical steady-state results are presented after 186 steps. The maximum step tolerable has not been determined, although more recent computations appear reasonable when a step size of .03 is employed. However, catastrophic divergence occurs at step sizes of around 10 CFL. [The CFL (explicit) stability limit for the ADI mesh is approximately .023.] Thus, it would appear steady state could be reached in fewer steps, by consistently using the maximum tolerable step.

Velocity vector plots for the FEM flowfield, evaluated at triangle vertices, are presented in figure 4. To gain some idea of the comparative density of FEM and ADI grid points, the FEM flowfield is interpolated to the ADI grid using the cubic finite element model of the solution (see fig. 5).

Figure 6 shows steady state velocity and density variations at stations $x_1 = .075$ and $x_2 = .175$ in the flow. The pressure is constant at .00389 over the field, with maximum deviations of 5×10^{-5}.

Table 1 shows actual numerical differences between the final steady state FEM and ADI computations. Steady state was defined by the convergence criterion

$$\left| \frac{\Delta f_n}{f_n} \right| \leq .01 \quad ,$$

where $\Delta f_n = f_{n+1} - f_n$ is a time difference and $f = \rho$, u, or v. Table 2 and figures 7 to 9 present percent differences between the two calculations,

$$\% \text{ difference} = \left| \frac{\text{FEM} - \text{ADI}}{\text{ADI}} \right| \times 100$$

with ADI results as base. As expected, extreme differences occur in the mixing region, where boundary condition differences on the outflow have most effect.

Due to the fact that the normal velocity component was zero or near zero over relatively large regions of the field, lack of enough significant figures of accuracy contributes to misleading percent differences in data for this component unless this is taken into account. Therefore, in computing this data the percent difference was automatically set to zero if either

$$|v| \leq 5 \times 10^{-5}$$

or the actual difference satisfied

$$|FEM - ADI|_v < 10^{-5} \quad .$$

Thus, only four significant decimal places of accuracy are assumed in the FEM solution; this appears to be a realistic assumption, from the manner of approach to steady state.

The relative rates of convergence for the two methods are shown in figure 10, which exhibits transient behavior of the fluid dynamics variables of selected field points above, below, and within the shear layer. The time to convergence for the two methods appears to be almost identical, with the FEM algorithm exhibiting the less smooth approach to steady state. This we possibly may attribute to the non-uniform time step and non-uniform mesh employed by FEM, versus the constant time step and uniform mesh used for ADI [the ADI calculation consistently employed a time step of (CFL =) .02337].

Cost Comparisons

The ADI code requires machine core storage of 107 K_8 for the 1220 node (10 x 122) mesh (this core requirement would change little had the truncated domain been used in the computation). For the truncated problem, the FEM code requires 236 K_8 words of storage. Moreover, using the full ADI domain would require approximately 20 percent more core, since the FEM core size is problem dependent. The CDC-6600 CPU time for ADI is .00374 sec/node and FEM requires .229 sec/node, for each time step. Additionally, the FEM code utilizes .8 sec/node of PPU time per step, for disk reading of time invariant data (which otherwise would require additional CPU time per step) and disk buildup of system matrices zR, Rz.

To provide some idea of where FEM CPU time is used, normal data printout and assembly of system matrices requires approximately 86.5 percent and equation solving 13.5 percent of the time used per step. Since over 1800 linear equations characterized by large bandwidth nonsymmetric matrices are being solved per step, it appears any inefficiency of the code occurs in that portion devoted to equation assembly.

Alternate Program Design

In restructuring the program design of the finite element code produced under contract NAS1-11707-37, two versions of code which implement the basic numerical algorithm and which provide greater mesh refinement capability have been produced. Version I incorporates some minor design alterations and program optimization performed by personnel of Computer Sciences Corporation, and is structured so that only one of the matrices D, zz, zR, Rz, RR occupies core at one time. As equation assembly proceeds the density and velocity local stiffness matrices for each element are assembled in parallel. In turn each global matrix is read into core from disk, local equation contributions are globally distributed, and the matrix is written back. As a result of these huge data volume disk writes (say, 20,000 words per write), peripheral processing time is exorbitant, and the result is extremely poor throughput, caused by excessive roll-outs of the program due to its heavy demands on system resources. For example, in one run preceding a moderate number of time steps, day file entries stretched over a twelve-hour period, with around 15 roll-outs, although total dollar cost quoted in the day file was around \$200. (PPU time carries no charge on this operating system other than through o/s calls.)

Design philosophy of Version II, programmed by the authors, centered around the goal of minimum core requirements as well as minimum data volume at each disk write. The outcome is greatly improved throughput, due to decreased data volume per disk write (a maximum of 220 words/write except for a few isolated large volume writes per step), at the expense of a greater number of o/s calls and slightly greater core requirements. In version II continuity assembly and equation solution is disposed of, without disk storage of any data involving the D-matrix, prior to commencing assembly of the momentum equations. During momentum assembly, the matrices zz and RR are built up in core, with local element stiffness matrices for zR and Rz written to disk triangle by triangle, each record of length 200 words. In-core global matrices are then written to disk, element stiffness matrices are read, and zR, Rz are assembled in the space vacated by zz, RR. The global matrices zR, Rz are then written to disk by rows, one record containing one row each from zR and Rz. zz and RR are then read from disk (an isolated large volume disk operation) and during equation solution the rows of zR, Rz are read as needed.

A comparison of the relative merits of the two versions is provided below with reference to the 228 triangle mesh. It should perhaps be remarked that the data below is approximate, based on a few runs of duration four time steps each.

<u>o/s Calls</u>. Version II requires twice as many as version I, at approximately .135/node per step, on a normal run. Depending upon the number of programs in the system simultaneously competing for the same disk device, PPU time for version II can vary by as much as 200 percent, for the same number of time steps.

<u>Core storage</u>. 165 K_8, version I, as opposed to 236 K_8, version II.

Throughput. Excessive roll-outs for version II, around 10 to 1 compared to version II. This can mean an extra day of elapsed time in getting the program back to the programmer, at some times two or three days turn around time (without priority).

Dollar Cost. Total cost indicated in the day file shows version II twice as dollar-wise expensive, due basically to increased core and twice the number of o/s calls.

CPU Time. Approximately the same.

PPU Time. Approximately three times as much for version I as for version II.

As a result of program throughput and excessive demands on system resources of version I, version II was chosen, after conference with personnel of the Analysis and Computation Division at Langley Research Center, as the vehicle for steady state numerical solution of the free shear flow problem.

CONCLUSIONS

Under contract NAS1-11707-37 and NGR 1098 a finite element algorithm for solution of fluid flow problems characterized by the two-dimensional Navier-Stokes equations has been developed. Proof of concept was provided by the calculation of primitive flow variables for the free shear flow problem, which provided excellent numerical results in comparison to the ADI method. Unfortunately, the algorithm places heavy demands on computer system resources, in terms of CPU time, core storage, PPU time, and o/s calls. Thus, total dollar cost of executing the algorithm appears extravagant. Moreover, due to algorithm complexity as well as the host of data manipulation and supporting programs essential in minimizing human labor, development time is much in excess of that normally expected in finite difference applications.

The critical question in evaluating the method is how much performance improvement might be expected by alternate program designs alternate analytical formulations, such as the use of a simpler element. Versions I and II of the program give an indication of how simple trade-offs in system resources supply drastically different performance in terms of dollar cost opposed to less excessive demands on system peripheral devices and improved program throughput.

Considering the present program design, it appears that the method used for solving the large systems of equations that arise is by far more economical than other known direct or indirect linear system solvers for equation classes as general as those presently involved. Moreover, the total I/O design of the FEM code as well as economy in core storage through use of disk devices is enabled by the solver features which allow system matrices to be processed by parts during equation assembly and solution.

However, one might reasonably expect that alternate program designs or other analytical formulations could yield further economy in terms of equation assembly time. For example, one factor that stands out as a contributor to inefficiency is the 16-point quadrature scheme used for evaluation of integrals over a triangle. Theory predicts the quadrature should be exact for two-dimensional polynomials of total degree six in order not to vitiate the accuracy of the method achievable from cubic elements (ref. 11). Further theory predicts the existence of eight or nine point quadrature schemes possessing this degree of accuracy (ref. 13). Unfortunately, mathematicians have yet to synthesize such a scheme; research is lagging practical needs in this area. Thus, one may perceive significant improvement in algorithm performance could such schemes be found, since there is abundant use of quadratures throughout the code.

By way of alternate analytical formulations, there exist cubic elements on rectangles (ref. 14) which could equally well replace the present cubic elements on triangles. Moreover, 9-point quadrature schemes for rectangles of the necessary degree of accuracy do exist, in practice as well as in theory. One might fault the general applicability of rectangular elements for regions with curved boundaries; however, this seems an irrelevant point since fairly good isoparametric rectangular elements are known.

On the other hand, one might conjecture that for a problem of the present complexity, the simpler should be the approach to its solution. Hence, linear elements come into consideration. The matrices arising could be expected to be of larger dimension, but bandwidths should be significantly decreased; one could predict that equation storage would be somewhat comparable. Moreover, at lower degrees of accuracy such as is needed with linear elements, more efficient quadrature schemes are available.

In summation, the finite element method developed appears to have excellent prospects in fluid mechanic applications, with respect to convergence to the steady solution and accuracy of the final results. In terms of complexity of implementation and computer resource demands and costs, as presently formulated the finite element method does not compete with standard finite difference techniques. A critical evaluation of the applicability of the method in fluid mechanics would require one to decide how much more favorable would be alternate program or analytic design in the method of implementation, as well as a determination of what particular types of problems presently difficult for the finite difference approach might readily yield to finite elements, such as high Reynolds number flows and complex geometry problems.

APPENDIX A

A BLOCK ITERATIVE LU SOLVER WEAKLY COUPLED FOR LINEAR SYSTEMS

INTRODUCTION

In some fluid dynamic applications of the finite element procedure the governing systems of linear equations arising from the numerical analysis are weakly coupled between distinct sets of flow variables. Solving such systems by the usual fixed band algorithms results in excessive program execution times, in particular for large nonsymmetric matrices and the time implicit asymptotic approach to steady state solution. However, alternative equation solvers may be developed which exploit the weakness of the coupling to produce significantly decreased equation solving time.

Assuming such equation solving techniques are applicable, the natural implicitness of the finite element method becomes less of a setback; the method then appears more competitive with the well established finite-difference techniques. Moreover, such equation solvers could be applied in finite difference settings as well, in which case higher-order-accurate implicit numerical schemes which for efficiency do not rely on the tridiagonal systems arising from ADI (ref. 8) and spline interpolation methods (ref. 18) could evolve.

In this appendix a technique which shall be called the block iterative LU solver is outlined. The efficiency of the method is dependent upon the supposition of a weakly coupled linear system; hence, its lack of generality in application does not grant a cure-all for the large non-sparse nonsymmetric system problem. However, where weak coupling is present the execution speed of the block solver makes it an opportune method for solving large linear systems. Furthermore, such systems appear frequently enough in scientific applications to warrant some general attention being devoted to this hybrid technique.

Consider the linear system

$$A\bar{Z} = \bar{b} \quad , \tag{A-1}$$

where A is an n-square matrix, with $\bar{Z}$ and $\bar{b}$ n-dimensional vectors. Partitioning the vector $\bar{Z}$ to obtain a vector $\bar{X}$ which has p components and a vector $\bar{Y}$ with q components, $p + q = n$, induces the partitioned system:

$$\left[\begin{array}{c|c} B & C \\ \hline D & E \end{array}\right] \left[\begin{array}{c} \bar{X} \\ \hline \bar{Y} \end{array}\right] = \left[\begin{array}{c} \bar{b}_1 \\ \hline \bar{b}_2 \end{array}\right] \tag{A-2}$$

Definition 1. The system (A-2) is said to be weakly coupled between the $\bar{X}$, $\bar{Y}$ sets of variables if the quantities

$$\begin{aligned} c_x &= //B^{-1}// \;\; //C// \\ c_y &= //E^{-1}// \;\; //D// \end{aligned} \tag{A-3}$$

are small compared to unity.

Here we use the usual Euclidean norm $//\bar{X}//_E$ for vectors, and the compatible matrix norm

$$//M// = \max_{//X//_E = 1} //M\bar{X}//_E \quad . \tag{A-4}$$

Since

$$c_x \geq \frac{//C//}{//B//} \qquad \text{and} \qquad c_y \geq \frac{//D//}{//E//} \tag{A-5}$$

the essential content of this definition is the intuitive idea that the system (A-2) is weakly coupled provided the elements of the matrices C, D are small compared to those of B, E.

DESIGN OF THE BLOCK ITERATIVE SOLVER

Consider the following equations defining an iterative solution of equation (A-1):

$$\begin{aligned} B\overline{X}^{k+1} &= \overline{b}_1 - C\overline{Y}^{k} \\ E\overline{Y}^{k+1} &= \overline{b}_2 - D\overline{X}^{k} \end{aligned} \qquad k = 1, 2, 3, \ldots n \qquad \text{(A-6)}$$

where B, E are assumed invertible. A sufficient condition for convergence of the iteration is that the coupling coefficients C_x, C_y of (A-3) each be less than unity; that is, the condition required for weak coupling.

Compared to direct methods of solution, the efficiency of this iteration is dependent upon the sparseness of C and D; the method of inversion of B, E; the closeness of the initial vector to the true solution of the system; and the weakness of the coupling. Of these, the method of inverting B, E is usually the only controllable influence. The block iterative LU solver proposed here is thus to be comprised of the block Jacobi iteration (A-6) using an LU decomposition for the inversion (once) of B, E, with subsequent front and back solves at each iterative step. (Of course, the matrices B, E must be sufficiently diagonal dominant to allow the LU decomposition.)

A tandem advantage of such a solver, in addition to speed considerations, is that for large systems the problem splits naturally into pieces which may be stored in core or on external I/o devices. For example, the matrices B, E may be retained in core, and C, D row stored on disk, to be read as needed when forming the right members of (A-6). For large systems this factor alone could dictate the choice of the hybrid solver.

PROGRAMMING RESULTS FOR A SPECIFIC APPLICATION

Weakly coupled linear systems of the form (A-2), with $p = q = \frac{n}{2}$, arise in certain time implicit finite element models for solution of the two-dimensional viscous compressible Navier-Stokes equations in primitive variables, such as the present, where the assumption of constant total temperature removes the necessity of solving the energy equation. Here system (A-1) represents the finite element modelled momentum equations, and $\overline{X}$, $\overline{Y}$ are the collections of u, v velocity variables associated with mesh points. The finite element algorithm is such that elements of the C, D matrices contain a factor of timestep divided by Reynolds number, τ/Re, not included in the elements of B, E. Weak coupling results from small time steps and high Reynolds number flows, say $\tau = 0(10^{-3})$ and $Re = 0(10^{6})$, in which case experimental observation shows the elements of B, E are $0(10^{-3})$ compared to C, D elements, which are $0(10^{-8})$. Hence on a 14-digit machine such as the CDC-6600 the elements of C, D are definitely significant in the problem solution, but the cross coupling between the two sets of velocity variables is rather weak.

All factors contributing to efficiency in execution of the block solver are present in the problem discussed. Finite element matrices are naturally sparse, and in the present case nonsymmetric and variable in bandwidth. The coupling is very weak, and the initial vector is the flow field one small timestep removed from the final solution vector, assuring rapid convergence. The B, E matrices are small perturbations from symmetric, positive definite matrices, so the LU decomposition applies.

The test case presented in the present paper is characterized by a system of $n = 1086$ equations, with the square matrices B, C, D, E (zz, zR, Rz, RR) nonsymmetric and variable band, but with all having identical intraband distributions of zero and non-zero elements. All matrices were profile stored (ref. 17), with space allowed for matrix fill in the decomposition of B, E.

The LU decomposition algorithm took no advantage of matrix sparseness other than knowledge of profile (fill and variable bandwidth). The semi-bandwidth* of each of B, C, D, E, if considered as fixed band matrices, was 24. Thus, the semi-bandwidth for the composite system (A-1) would be 567. The matrices B, E were stored in-core, with C, D row stored on disk, one disk record being comprised of one row from C and one row from D. To solve such a system on the CDC-6600 computer required approximately 12 seconds CPU time and 8 iterations to convergence initially, and approximately 11 seconds and 7 iterations after the time asymptotic solution had progressed several steps (showing the influence of the initial vector). Iterative improvement (ref. 5) experimentation indicates 12 significant figures of accuracy (on the 14-digit CDC-6600 series) for the solution.

* A matrix of bandwidth $2m + 1$ has semi-bandwidth m(integer).

APPENDIX B

A MESH GENERATION PROGRAM (MESHGN)

INTRODUCTION

Manual generation of a finite element mesh and mesh associated data is most time consuming; for large grids the task becomes nearly impossible to accomplish accurately. For example, although modern computers can solve a plane stress problem (steady state) with a thousand elements in, say, under ten minutes, the division of the field into elements and the associated data preparation and checking may take several days. Moreover, one has no guarantee that the first trial run with the finite element numerics model will not uncover some irregularity in solution behavior requiring mesh refinement in portions of the mesh, if not over the whole mesh, requiring the job be repeated! Thus, an essential component for success of any finite element endeavor is the availability of an automatic mesh generation program.

To overcome this obstacle, an automatic mesh generation routine has been programmed. Since the problem domain is rectangular, the structuring of such a program of this code appeared more feasible than obtaining and becoming familiar with the intricacies of other mesh generators on the market.

The code developed triangulates a rectangular region, numbers the nodes associated with triangle vertices and centroid, and generates mesh-associated data arrays. Use of the code reduces mesh generation manual labor requirements to about two hours, for a large mesh having above 600 unknowns per dependent variable.

Mesh Geometry

The routine superimposes a rectangular grid work on the problem domain, as in finite differences, then divides each rectangle into triangles (see fig. 3). In the numerical evaluation of boundary integral quadratures within the finite element algorithm, it is assumed that no triangle has more than one edge on the boundary of the region. Therefore, top right and bottom left rectangles are subdivided into triangles whose hypotenuse is the lower left to top right diagonal of the rectangle. The reverse situation applies for all other rectangles. The finite difference rectangles are formed by mesh intersections of the lines $x = x_i$, $i = 1, 2, \ldots$ NXG and $y = y_j$, $j = 1, 2, \ldots$ NYG. Some degree of irregularity in the mesh may be achieved by non-uniform spacing of the x_i, y_j subdivisions. For proper functioning of the routine the minimum number of triangles that can be configured is eighteen; i.e., $\overline{\text{NXG}}$ and $\overline{\text{NYG}}$ must each exceed three.

MESH NUMBERING

Triangle Numbering

Triangles are numbered left to right and top to bottom, IT = 1, 2, ... , M. Here IT designates triangle number, and M the total number of triangles.

Node Numbering

The spatial variation of each dependent physical variable in the flow field is approximated by a finite element trial function. Each trial function contains parameters approximating flow variable function and first partial derivative values at triangle vertices are referred to as <u>multiple nodes</u>, having three problem nodes associated with each. Triangle centroids are <u>simple nodes</u>. Nodes are numbered left to right and top to bottom, a level of multiple nodes followed by a level of simple nodes (see fig. 3).

The two-dimensional array NODE(M, ID3), M = the total number of nodes, indicates which geometrical nodes are associated with a particular triangle. Node numbers for triangle IT are stored in NODE(IT,J), J = 1, 2, ..., 10. These ten node numbers are obtained by proceeding counterclockwise around each triangle, procuring sequentially geometrical node numbers previously assigned, with the centroidal node procured last. (For meaningful assembly of the finite element equations, it is essential that a consistent choice of either counterclockwise on all triangles, or clockwise on all, be made.) Normally the starting vertex for a particular triangle is of no consequence. However, in order that boundary integrals in the finite element numerics be properly computed, the present program assumes the starting vertex for triangles with one edge on the boundary be interior to the problem domain.

Variable Numbering

Node numbers just discussed refer to a physical location in the problem domain. Now consider in turn each particular physical dependent flow variable, such as density or a velocity component. There is associated with each node a parameter of the trial function which approximates this flow variable. If a boundary condition applies at this node, the parameter is a known variable; otherwise an unknown. Each category of variables is numbered separately, and these numbers do not necessarily coincide with node numbers.

Unknown variable numbering. Proceeding triangle by triangle, and counterclockwise around the nodes of each, the mesh generator assigns sequentially unknown variable numbers of nodes where no boundary condition applies, provided the node has not been numbered already on a previous triangle. To minimize matrix fill in the resulting system of finite element equations, the unknown variable numbers are then reverse ordered, as in the Reverse-Cuthill bandwidth minimization routine. It has been verified that this process gives near minimal profile for the system of equations. For example, the bandwidth minimization program of Poole (ref. 15) did not yield as economical a matrix storage as the present process, although it produced slightly smaller maximum bandwidth.

Known variable numbering. The only input to the mesh generator as far as variable numbering is concerned is a list of the nodes at which boundary conditions apply. These nodes are sequentially assigned known variable numbers. The input list of nodes need have no particular order. Of course, a separate node list for density and velocity boundary data is required.

Variable number access. The array IØR(2, ID3) establishes the correspondence between known or unknown variable and the physical node with which the variable associates. For example, the variable number associated with node I is obtained from the FORTRAN statement

```
IV = IØR(IP, I)   .
```

For IP = 2, IV is a velocity variable number; for IP = 1, it is the number of a continuity variable.

Variable number discrimination. The discrimination of whether a variable number access yields the number of a known or an unknown variable is determined from bit settings in the array FLAG(M). If the node associated with the variable is on triangle IT, the bit settings of the location FLAG(IT) determine whether the variable is known or unknown. The bit settings in location IT of the array FLAG in general display information concerning triangle IT and variables associated with it.

Slack variables. Slack variables, which are defined below, are employed in the velocity components to assure that velocity variables associated with a particular node are both known or both unknown. For example, if one component is known and the other unknown, the equation

$$x = \text{known value}$$

is assembled for the known variable, and it is otherwise treated as unknown throughout the equation assembly and equation solving process. This avoids separate variable lists between velocity components, and leads to determining equations for momentum variables identical in profile for the two sets of momentum matrices. Bookkeeping in the equation solving is greatly simplified. Throughout mesh generation it is automatically assumed that velocity variables at a node are both known or both unknown; node numbers of slack variables are input to the finite element numerics program, and after equation assembly the proper equations are modified to produce slack variables.

Triangle Coordinates

Triangle coordinates are created by MESHGEN and stored in an array CØØRD (M,2,4); x-y coordinates of all vertices and the centroid. This three-dimensional array is not used in the finite element numerics algorithm directly; coordinate associated quantities such as the Jacobian of the affine mapping which transforms a particular triangle onto the standard triangle with vertices (0,0), (1, -1), and (1,1) (used in evaluating numerical quadratures) are computed and passed along.

MESH ASSOCIATED DATA ARRAYS

The arrays NODE, COORD, FLAG, IOR discussed in the preceding section, automatically generated by MESHGEN, are the output which one would normally ascribe to a mesh generator. Before MESHGN was coded, the manual generation and checking of these arrays, for a 103-triangle mesh, required over a week, with no guarantee of accuracy. The availability of this data permits computation of other mesh associated data discussed in the sequel, such as band structure and storage information for system matrices.

In-core Storage of System Matrices

As previously indicated, the finite element system matrices are the density matrix, D, and the velocity matrices zz, ZR; Rz, RR. These matrices are variable band, of significant maximum bandwidth, sparse within the band, and nonsymmetric. However, all matrices have symmetric profile; i.e., if matrix element

a_{ij} is nonzero, so is a_{ji}. The use of slack variables forces all velocity matrices to be of the same dimension and have identical profile, leading to simpler bookkeeping. The profile matrix storage scheme used accounts for the variable bandwidth, but does not utilize intra-band matrix sparseness.

Let $\overline{R}_i$ be a vector whose elements a_{ij} are the elements of a matrix A_α which occur in row i, starting with the first nonzero element and ending with the last nonzero element. The matrix is then stored as a one-dimensional array by stacking end-to-end the vectors $\overline{R}_i$.

Matrix elements of D, zz, RR are accessed by storing locations of the diagonal elements in the array JT(2,N) N the maximum dimension of D, zz. The code

```
IP = JT(I,K) + J - K
```

obtains the location of element a_{KJ} of D, for I = 1, and the corresponding element location in zz,RR, if I = 2.

In performing the LU decompositions of D, zz, RR it is necessary to know the left and right semi-bandwidths, or the number of elements in a matrix row to the left and right of the diagonal. This information is stored in the arrays IBS(2,N) and IBL(2,N). Here

```
IP = IBS(I,J)
```

accesses density information if I = 1 and velocity information if I = 2. This convention holds for arrays IBS, IBL, JT.

The mesh generator computes the maximum bandwidth for each matrix row, adjusts the bandwidth to allow for any matrix fill occurring in the decompositions of D, zz, RR, and computes diagonal element locations (JT) and semi-bandwidth (IBS,IBL) information reflecting allowance for matrix fill.

Disk Storage of System Matrices

Two optimized versions of the original finite element numerics program produced under contract NAS1-11707-37 have been programmed. Version I, consisting of some minor design alterations and program optimization by personnel of Computer Sciences Corporation, is structured so that only one of the matrices D, zz, zR, Rz, RR occupies core at one time. As equation assembly proceeds, the density and velocity local stiffness matrices for each element are assembled; each partially assembled global matrix is read into core in turn; the local equation contributions are globally distributed; and the matrix is written back to disk. As lower semi-bandwidths for each row of zz, zR, Rz, RR are identical, the IBS array as described above serves all. However, since no matrix fill in Rz, zR need be accounted for in the equation solving, upper semi-bandwidths may differ between rows of zz, RR and zR, Rz. This necessitates the keeping of arrays JTV(N), IBLV(N), similar in structure to JT and IBL except for fill allowance, which indicates diagonal element storage and upper semi-bandwidth information for zR and Rz.

Version II, with design alterations by the authors, differs in structure from version I. Here continuity assembly and solution completes, with no disk storage of D, before momentum equation assembly starts. During momentum assembly, zz and RR are built up in-core, with local element stiffness matrices for zR, Rz written to disk, triangle by triangle, each record of length 200 words. In-core matrices are then written to disk, element stiffness matrices are read, and zR, Rz are assembled in the space vacated by RR, zz. The matrices Zr, Rz are written to disk by rows, each record containing one row each of zR and Rz. The arrays IBS, IBC, JT, IBLV above described apply in version II as well. However, since zR and Rz must be assembled en toto prior to any disk write, the JTV array now used locates only the diagonal element of zR, with the corresponding row of Rz stacked end-to-end with it. The row records of zR, Rz are read as needed in the block iterative solution of the momentum equations.

Time Invariant Data

In the finite element numerics program B-spline function and derivative values, as well as area integrals of combinations of such, are needed at each time step, but do not change with time instant. These B-spline quantities and time invariant integrals are computed, for each triangle, in the mesh generator, to be read from disk as needed in the time asymptotic computation.

REFERENCES

1. France, P.W., Lewis, R.W., and Norris, V.A., "Finite Element Analysis of the Motion of a Gas-Liquid Interface in a Porous Medium", Int. J. Num. Meth. Eng., Vol. 9, 433-448 (1975).

2. Cheng, Ralph T., "Numerical Investigation of Lake Circulation Around Islands by the Finite Element Method", Int. J. Num. Meth. Eng., Vol. 5, 103-112 (1972).

3. Fix, George J., "Finite Element Models for Ocean Circulation Problems", SIAM J. Appl. Math., Vol. 29, No. 3, 1975 (To appear).

4. Brashears, M.R., Chan, S.T.K., and Young, V.Y.C., "Finite Element Analysis of Transonic Flow", Proceedings, International Conference on Computational Methods in Nonlinear Mechanics, Huntsville, AL, September 1974.

5. Baker, A.J., "A Finite Element Solution Algorithm for the Navier-Stokes Equations", NASA CR-2391, Langley Research Center, Hampton, VA, June 1974.

6. Baker, A.J., "Computational Techniques for Pressure Distributions in Viscous Fluid Dynamics", Bell Aerospace Report No. 9500-920310, Buffalo, NY, December 31, 1973.

7. Rudy, D.H., Morris, D.J., Rubin, S.G., et al., "An Investigation of Several Numerical Procedures for Time Asymptotic Compressible Navier-Stokes Equations", Aerodynamic Analysis Requiring Advanced Computers, Pt. 1, NASA SP-347, 1975, pp. 457-468.

8. Roache, Patrick, J., Computational Fluid Dynamics, Hermosa Publishers, Albuquerque, NM, 1972.

9. Oh, Y.H., "Analysis of Two-Dimensional Free Turbulent Mixing", AIAA Paper No. 74-594, American Institute of Aeronautics and Astronautics, New York, NY, June 1974.

10. Cooke, C.H., and Fix, George J., "A Finite Element Solution Algorithm for the Two-Dimensional Compressible Navier-Stokes Equations", ICASE Report, Langley Research Center, Hampton, VA, November 29, 1973.

11. Fix, George, J., "Effects of Quadrature Errors in Finite Element Approximations of Steady State, Eigenvalue, and Parabolic Problems", Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations, Academic Press, New York, NY, 1972, pp. 525-556.

12. Hammer, P.C., Marlowe, O.P., and Stroud, A.H., "Numerical Integration over Simplexes and Cones", Math. Tables and Aids to Comp., 10, 130-137, 1956.

13. Stroud, A.H., Approximate Calculation of Numerical Integrals, Prentice-Hall, Englewood Cliffs, NJ, 1971.

14. Fix, G.J. and Strang, G., An Analysis of the Finite Element Method, Prentice-Hall, Englewood Cliffs, NJ, 1973.

15. Gibbs, N.E., Poole, W.G., and Stockmeyer, P.K., "An Algorithm for Reducing the Bandwidth and Profile for a Sparse Matrix", ICASE Report, Langley Research Center, Hampton, VA, July 1974.

16. Rubin, S.G., and Graves, Randolph, "A Cubic Spline Approximation for Problems in Fluid Mechanics", J. Comps. and Fluids, Vol. 3, pp. 1-36, 1975.

17. George, J. Alan., "Computer Implementation of the Finite Element Method", Ph.D. Dissertation STAN-CS-71-208, Stanford University, February 1971.

18. Forsythe, G.E., and Moler, Cleve, V., Computer Solution of Linear Algebraic Systems, Prentice-Hall, Englewood Cliffs, NJ, 1967.

TABLE I.- FEM-ADI STEADY STATE DIFFERENCES

Density, $\Delta\rho$			
x = 0.	.075	.175	.225
-0.000000	0.000000	0.000004	0.000006
-.000000	-.000000	.000003	.000005
-.000000	-.000000	.000002	.000004
-.000000	-.000000	.000002	.000004
-.000000	-.000000	.000002	.000005
-.000000	-.000000	.000003	.000006
-.000000	-.000000	.000004	.000007
-.000000	-.000000	.000004	.000008
-.000000	-.000000	.000005	.000009
-.000000	.000001	.000005	.000010
-.000000	.000001	.000006	.000011
-.000000	.000001	.000007	.000011
-.000000	.000001	.000007	.000020
-.000000	.000001	.000005	.000015
-.000000	.000001	.000011	.000016
-.000000	.000001	.000010	.000023
-.000000	-.000004	.000009	.000021
-.000000	.000006	.000017	.000029
-.000000	.000001	.000010	.000017
-.000000	.000011	.000008	.000003
-.000000	.000006	-.000010	-.000027
-.000000	-.000016	-.000027	-.000051
.000000	-.000030	-.000033	-.000044
-.000000	-.000021	-.000029	-.000036
-.000000	.000007	-.000005	.000001
-.000000	.000036	.000049	.000051
-.000000	.000020	.000050	.000062
-.000000	-.000003	.000014	.000022
-.000000	-.000005	-.000014	-.000009
-.000000	-.000004	-.000010	-.000009
-.000000	-.000005	-.000008	-.000002
-.000000	-.000004	-.000001	.000002
-.000000	-.000004	-.000004	-.000004
-.000000	-.000004	-.000004	-.000004
-.000000	-.000004	-.000004	-.000004
-.000000	-.000004	-.000004	-.000004
-.000000	-.000004	-.000004	-.000004
-.000000	-.000005	-.000005	-.000005
-.000000	-.000004	-.000004	-.000005

ORIGINAL PAGE IS
OF POOR QUALITY

TABLE I.- CONCLUDED.

$u_{FEM} - u_{ADI}$

Streamwise velocity, Δu			
x = 0.	.075	.175	.225
0.000000	0.000003	0.000000	0.000000
.000000	.000001	-.000014	-.000011
.000000	-.000007	-.000012	-.000030
.000000	-.000007	-.000023	-.000043
.000000	-.000008	-.000034	-.000063
.000000	-.000010	-.000047	-.000076
.000000	-.000012	-.000056	-.000092
.000000	-.000015	-.000064	-.000104
.000000	-.000016	-.000070	-.000115
.000000	-.000018	-.000075	-.000135
.000000	-.000020	-.000093	-.000144
.000000	-.000032	-.000109	-.000161
.000000	-.000035	-.000124	-.000184
.000000	-.000041	-.000140	-.000207
.000000	-.000035	-.000149	-.000240
.000000	-.000040	-.000167	-.000269
.000000	-.000042	-.000170	-.000317
.000000	-.000029	-.000158	-.000332
.000000	-.000014	-.000115	-.000270
.000000	.000001	-.000045	-.000086
.000000	.000002	.000017	.000177
.000000	-.000033	.000028	.000317
.000000	-.000040	.000003	.000266
.000000	-.000013	.000007	.000086
.000000	.000030	.000045	-.000030
.000000	.000036	.000069	-.000010
.000000	-.000007	.000022	.000036
.000000	-.000027	-.000038	.000006
.000000	-.000015	-.000030	-.000025
.000000	-.000009	-.000018	-.000016
.000000	-.000002	-.000005	-.000003
.000000	.000000	-.000002	.000002
.000000	-.000003	-.000002	-.000002
.000000	-.000004	-.000003	-.000003
.000000	-.000004	-.000004	-.000004
.000000	-.000004	-.000004	-.000004
.000000	-.000004	-.000003	-.000003
.000000	-.000004	-.000004	-.000004
.000000	-.000003	-.000003	-.000004

$v_{FEM} - v_{ADI}$

Normal velocity, Δv			
x = 0.	.075	.175	.225
0.000000	0.000000	0.000000	0.000000
.000000	.000000	.000000	0.000000
.000000	.000000	.000000	.000010
.000000	.000000	.000000	.000012
.000000	.000000	.000000	.000021
.000000	.000000	.000009	.000013
.000000	.000000	.000000	.000012
.000000	.000000	.000000	.000011
.000000	.000000	.000000	.000010
.000000	.000000	.000000	.000011
.000000	.000000	.000015	.000022
.000000	.000000	.000014	.000028
.000000	.000000	.000022	.000029
.000000	.000000	.000018	.000020
.000000	.000000	.000018	.000011
.000000	.000000	.000013	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000017	.000000
.000000	.000125	.000166	.000062
.000000	.000104	.000191	.000141
.000000	.000000	.000065	.000100
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000011	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000
.000000	.000000	.000000	.000000

TABLE II.- FEM-ADI PERCENT DIFFERENCE

Density, ρ				Streamwise velocity, u				Normal velocity, v			
x = 0.	.075	.175	.225	x = 0.	.075	.175	.225	x = 0.	.075	.175	.225
0.000	0.000	0.009	0.014	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
.000	.000	.007	.012	.000	.000	.002	.002	.000	.000	.000	.000
.000	.000	.005	.009	.000	.001	.002	.005	.000	.000	.000	.101
.000	.000	.005	.009	.000	.001	.004	.007	.000	.000	.000	.121
.000	.000	.005	.012	.000	.001	.006	.010	.000	.000	.000	.212
.000	.000	.007	.014	.000	.002	.008	.013	.000	.000	.000	.131
.000	.000	.009	.016	.000	.002	.009	.015	.000	.000	.000	.121
.000	.000	.009	.019	.000	.002	.011	.017	.000	.000	.000	.111
.000	.000	.012	.021	.000	.003	.012	.019	.000	.000	.000	.101
.000	.002	.012	.023	.000	.003	.013	.022	.000	.000	.000	.111
.000	.002	.014	.026	.000	.003	.015	.024	.000	.000	.152	.222
.000	.002	.016	.026	.000	.005	.018	.027	.000	.000	.141	.283
.000	.002	.016	.047	.000	.006	.021	.031	.000	.000	.223	.294
.000	.002	.012	.035	.000	.007	.023	.034	.000	.000	.183	.204
.000	.002	.026	.037	.000	.006	.025	.040	.000	.000	.185	.114
.000	.002	.023	.054	.000	.007	.028	.045	.000	.000	.136	.000
.000	.009	.021	.049	.000	.007	.028	.053	.000	.000	.000	.000
.000	.014	.040	.068	.000	.005	.026	.055	.000	.000	.000	.000
.000	.002	.023	.039	.000	.002	.019	.045	.000	.000	.000	.000
.000	.026	.018	.007	.000	.000	.007	.014	.000	.000	.000	.000
.000	.014	.022	.060	.000	.000	.003	.028	.000	.000	.000	.000
.000	.035	.058	.110	.000	.005	.004	.050	.000	.000	.000	.000
.000	.063	.068	.093	.000	.006	.000	.040	.000	.000	.088	.000
.000	.041	.055	.069	.000	.002	.001	.012	.000	.426	.685	.278
.000	.012	.010	.002	.000	.004	.006	.004	.000	.351	.770	.614
.000	.056	.077	.083	.000	.005	.009	.001	.000	.000	.325	.524
.000	.028	.072	.089	.000	.001	.003	.005	.000	.000	.000	.000
.000	.004	.019	.038	.000	.003	.005	.001	.000	.000	.000	.000
.000	.007	.019	.012	.000	.002	.004	.003	.000	.000	.000	.000
.000	.005	.013	.012	.000	.001	.002	.002	.000	.000	.000	.000
.000	.007	.010	.003	.000	.000	.001	.000	.000	.000	.000	.000
.000	.005	.001	.003	.000	.000	.000	.000	.000	.000	.000	.000
.000	.005	.005	.005	.000	.000	.000	.000	.000	.000	.000	.000
.000	.005	.005	.005	.000	.000	.000	.000	.000	.000	.000	.000
.000	.005	.005	.005	.000	.000	.000	.000	.000	.000	.000	.000
.000	.005	.005	.005	.000	.000	.000	.000	.000	.000	.000	.000
.000	.005	.005	.005	.000	.000	.000	.000	.000	.000	.000	.000
.000	.007	.007	.007	.000	.000	.000	.000	.000	.000	.000	.000
.000	.005	.005	.007	.000	.000	.000	.000	.000	.000	.000	.000

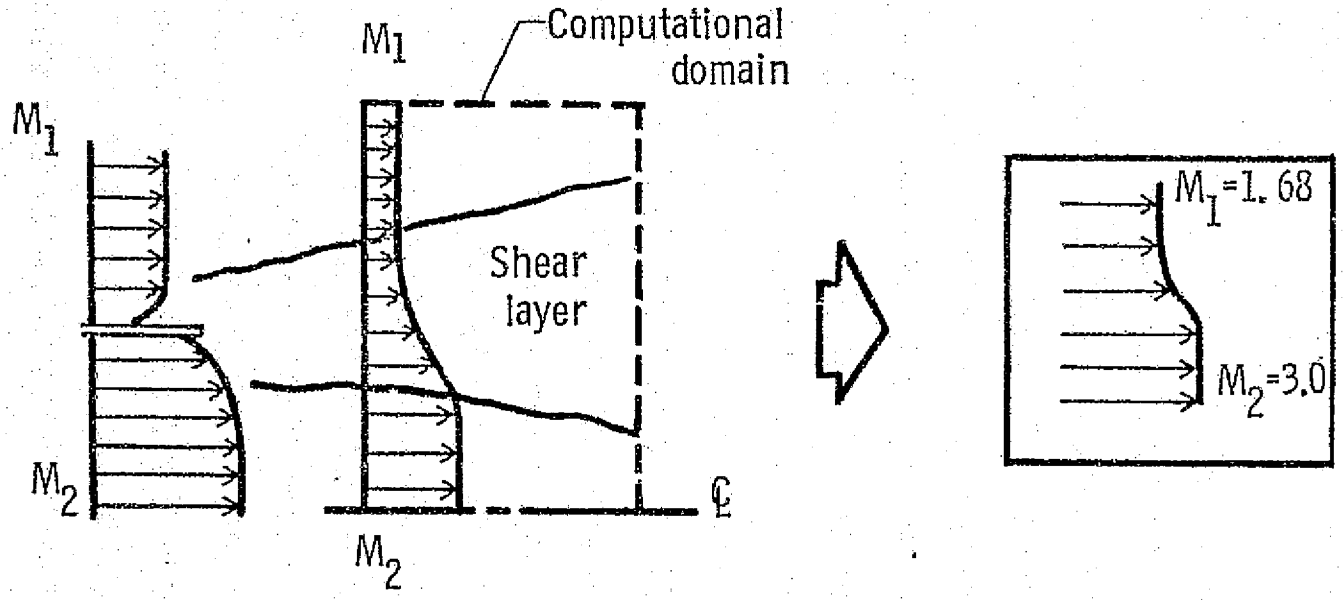


Figure 1. - Flow field configuration, supersonic jet mixing.

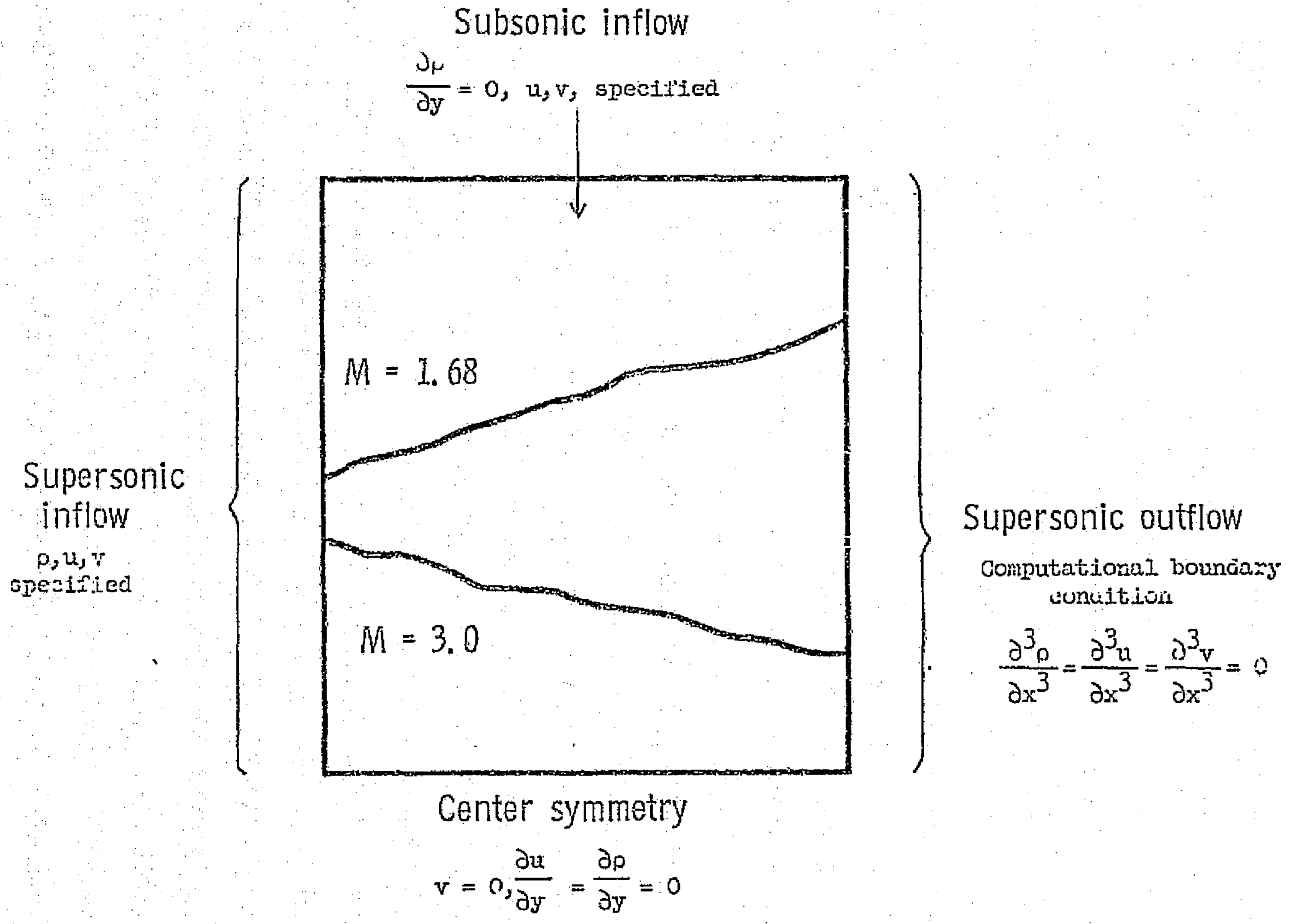

Subsonic inflow
$\frac{\partial \rho}{\partial y} = 0$, u, v, specified
M = 1.68
M = 3.0
Supersonic inflow
ρ, u, v specified
Supersonic outflow
Computational boundary condition
$\frac{\partial^3 \rho}{\partial x^3} = \frac{\partial^3 u}{\partial x^3} = \frac{\partial^3 v}{\partial x^3} = 0$
Center symmetry
$v = 0, \frac{\partial u}{\partial y} = \frac{\partial \rho}{\partial y} = 0$

Figure 2. - Boundary conditions on computational domain.

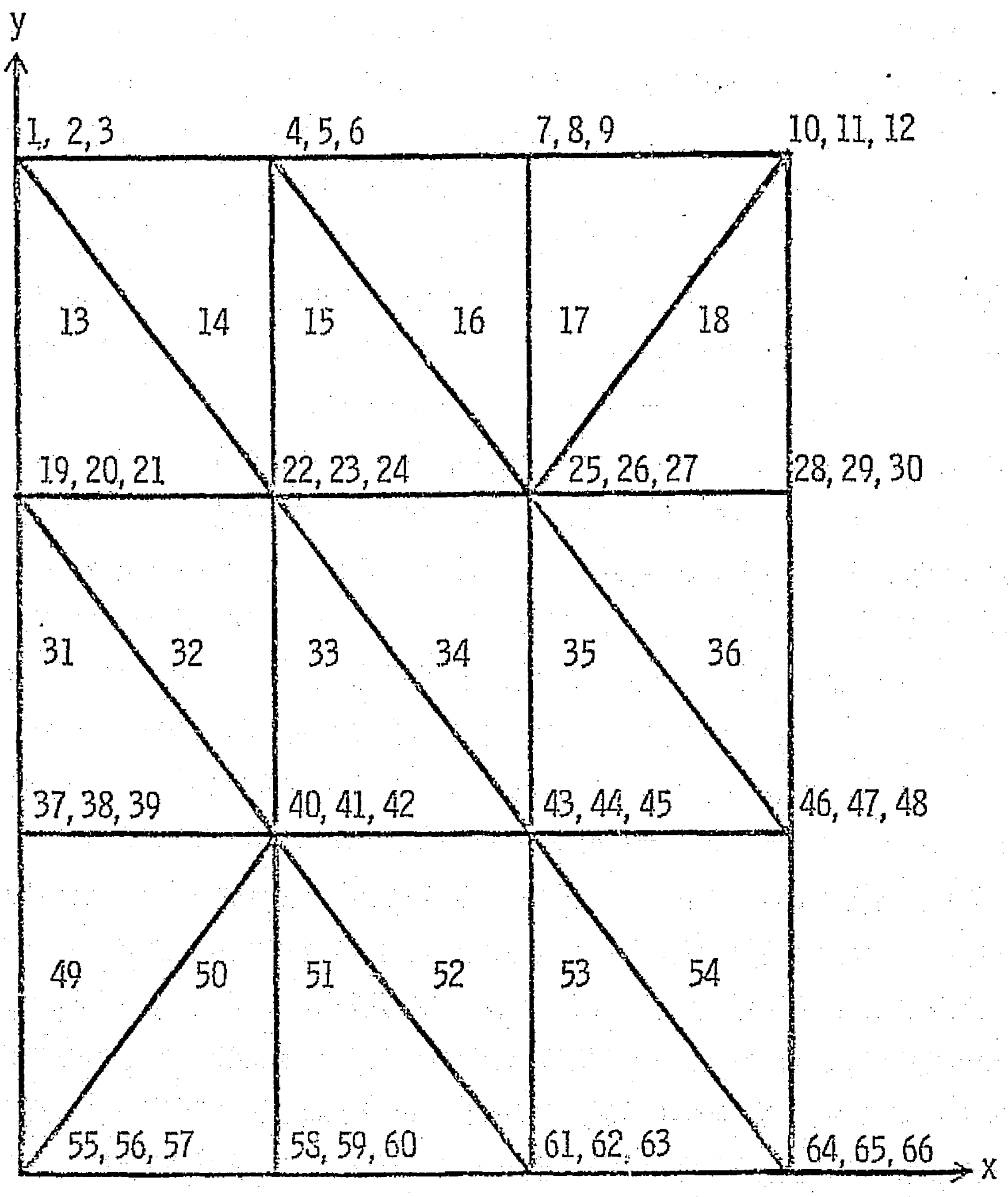


Figure 3. - Node numbering scheme for the B-spline element.

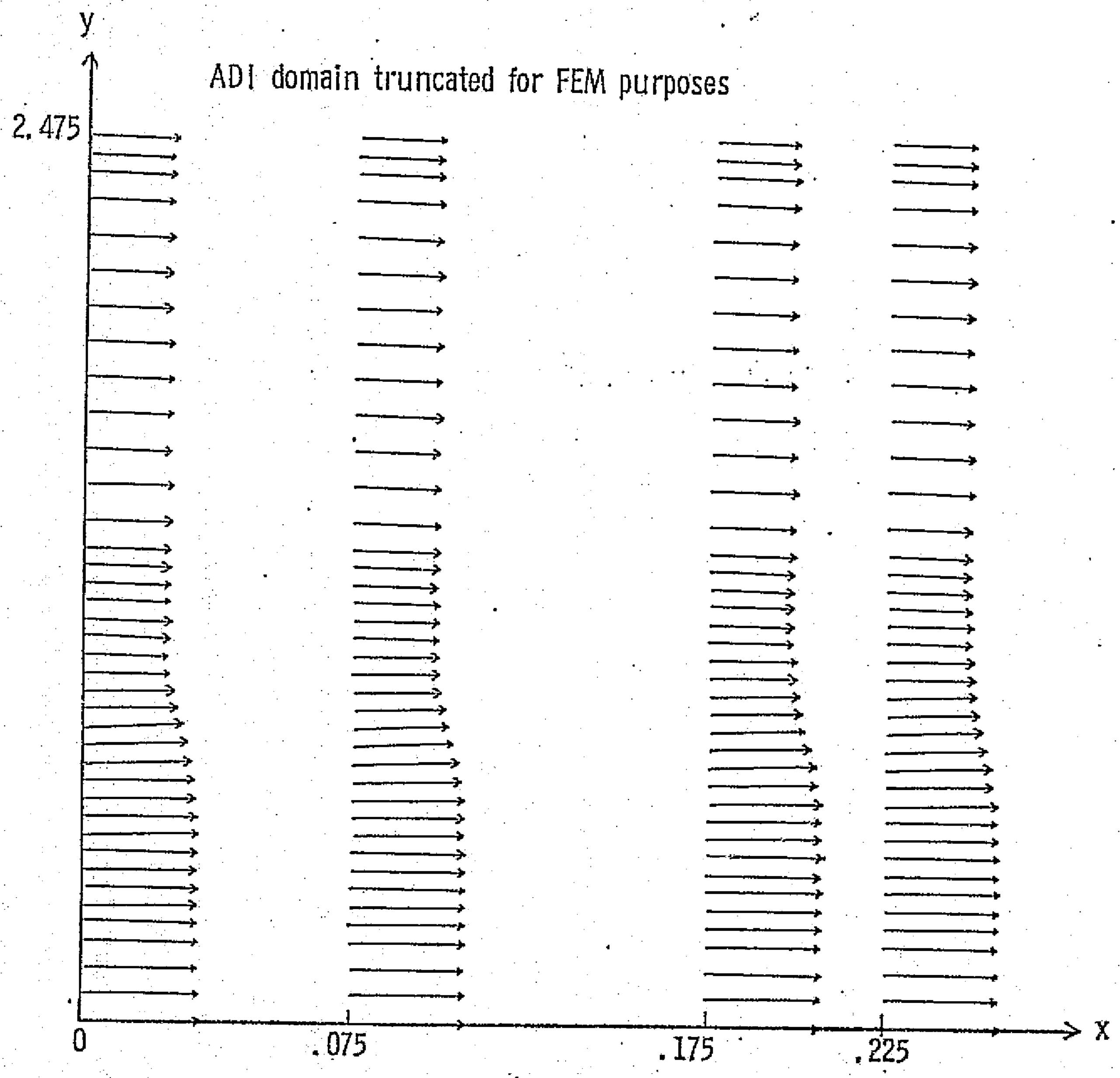


Figure 4. - Velocity vector plots for the FEM gridwork.

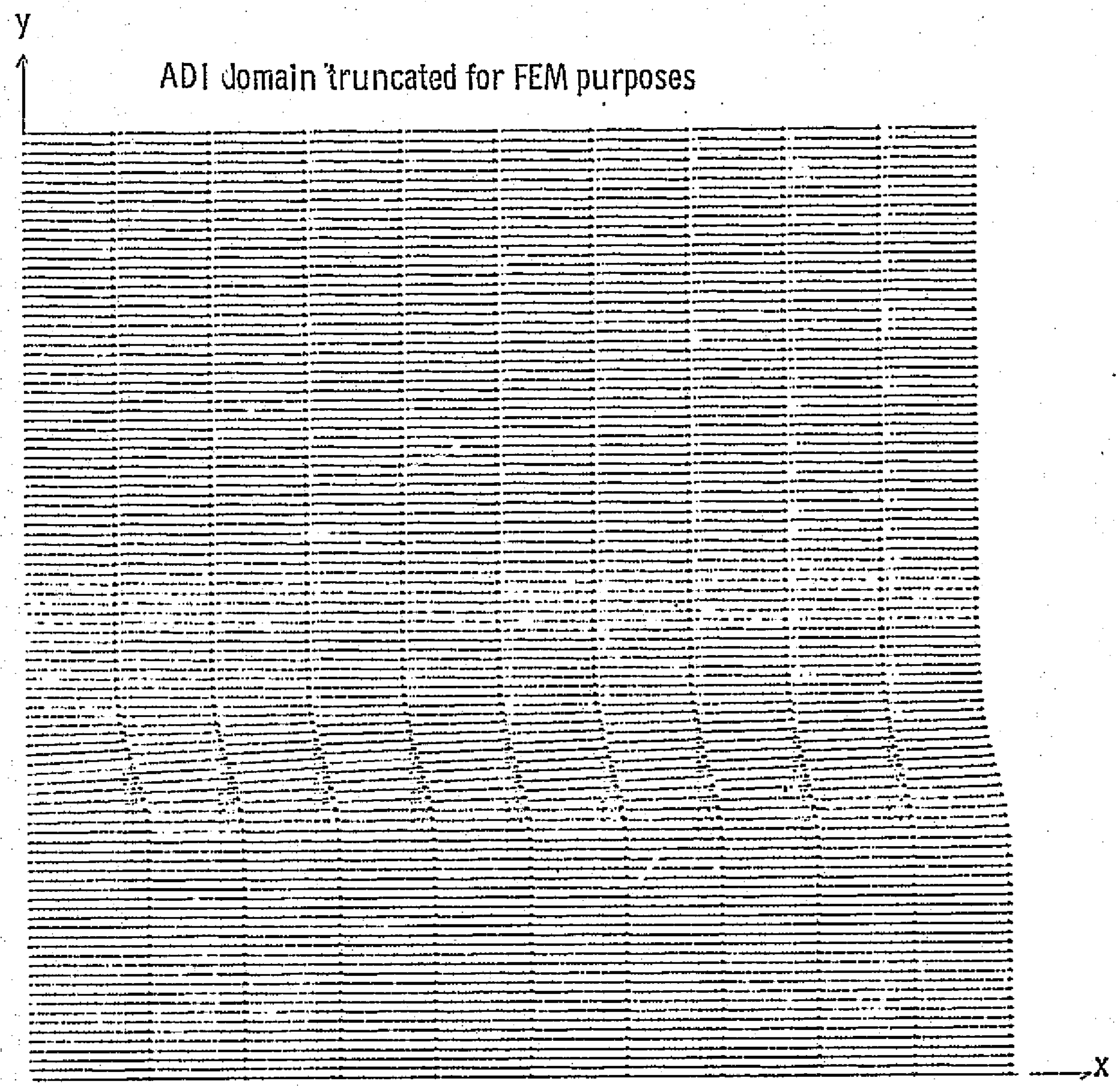
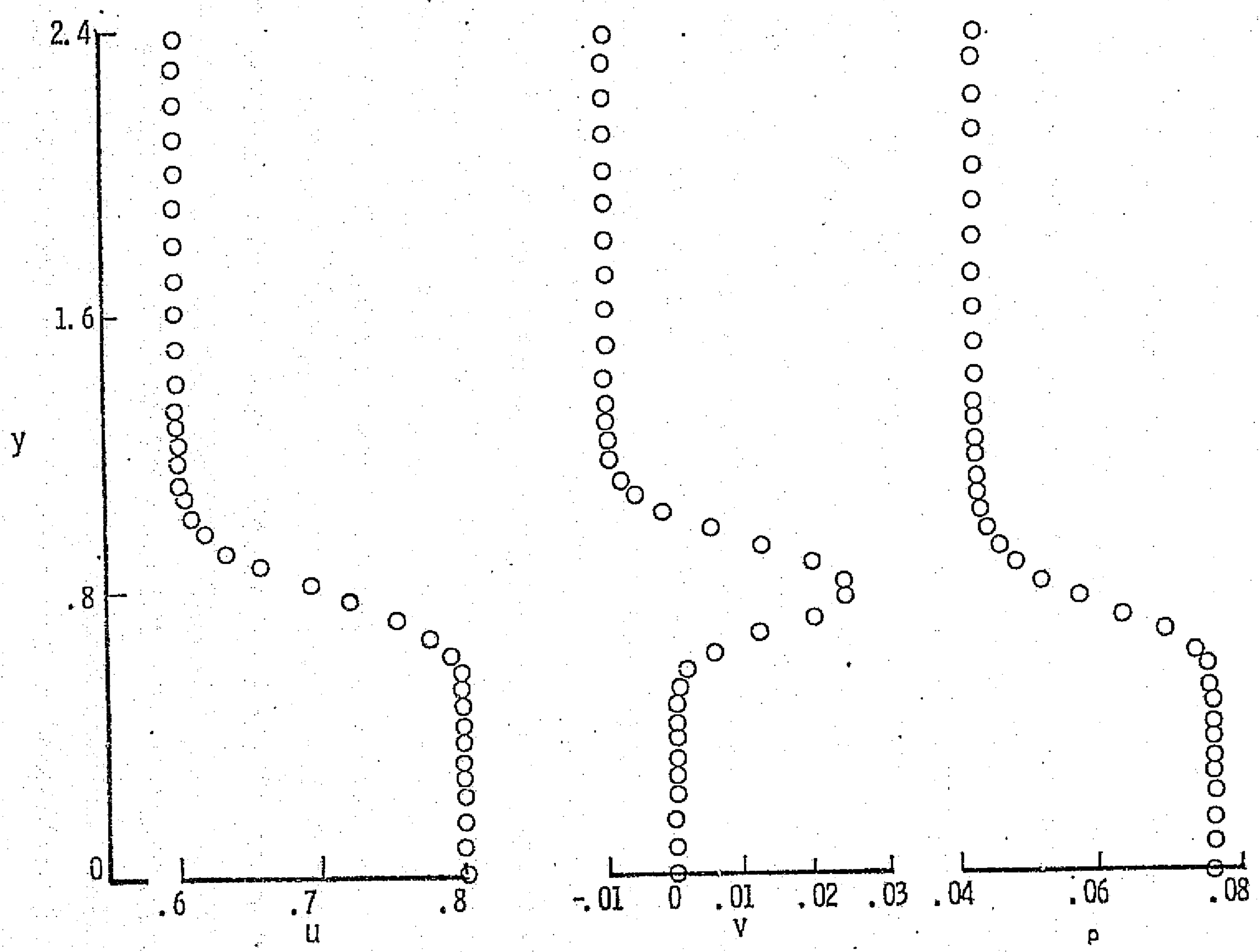
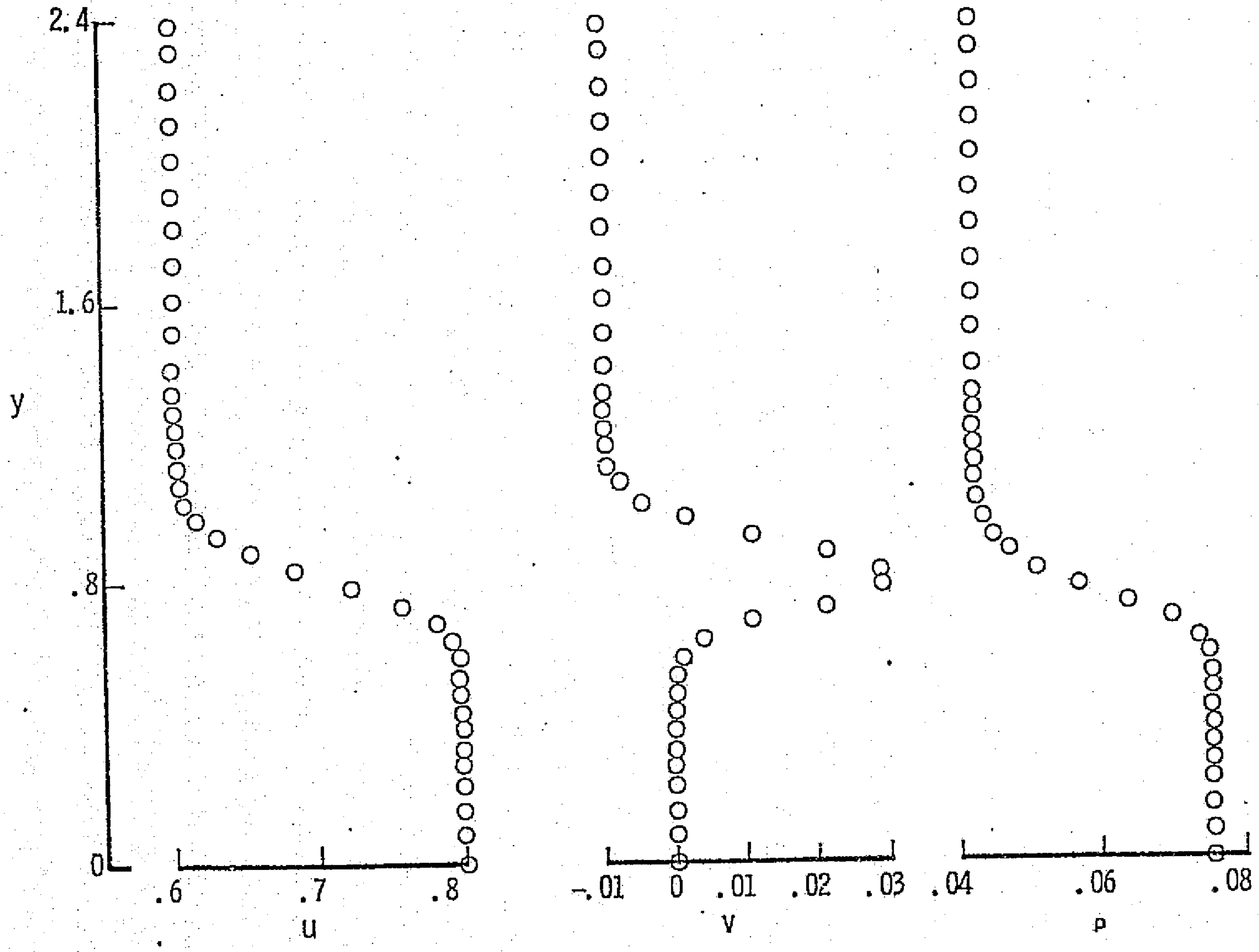


Figure 5. - Velocity vector plots for the ADI gridwork. (Interpolated using FEM solution model.)



(a) x = 0.075

Figure 6. - Steady state results, FEM solution.



(b) x = 0.175.

Figure 6. - Concluded.

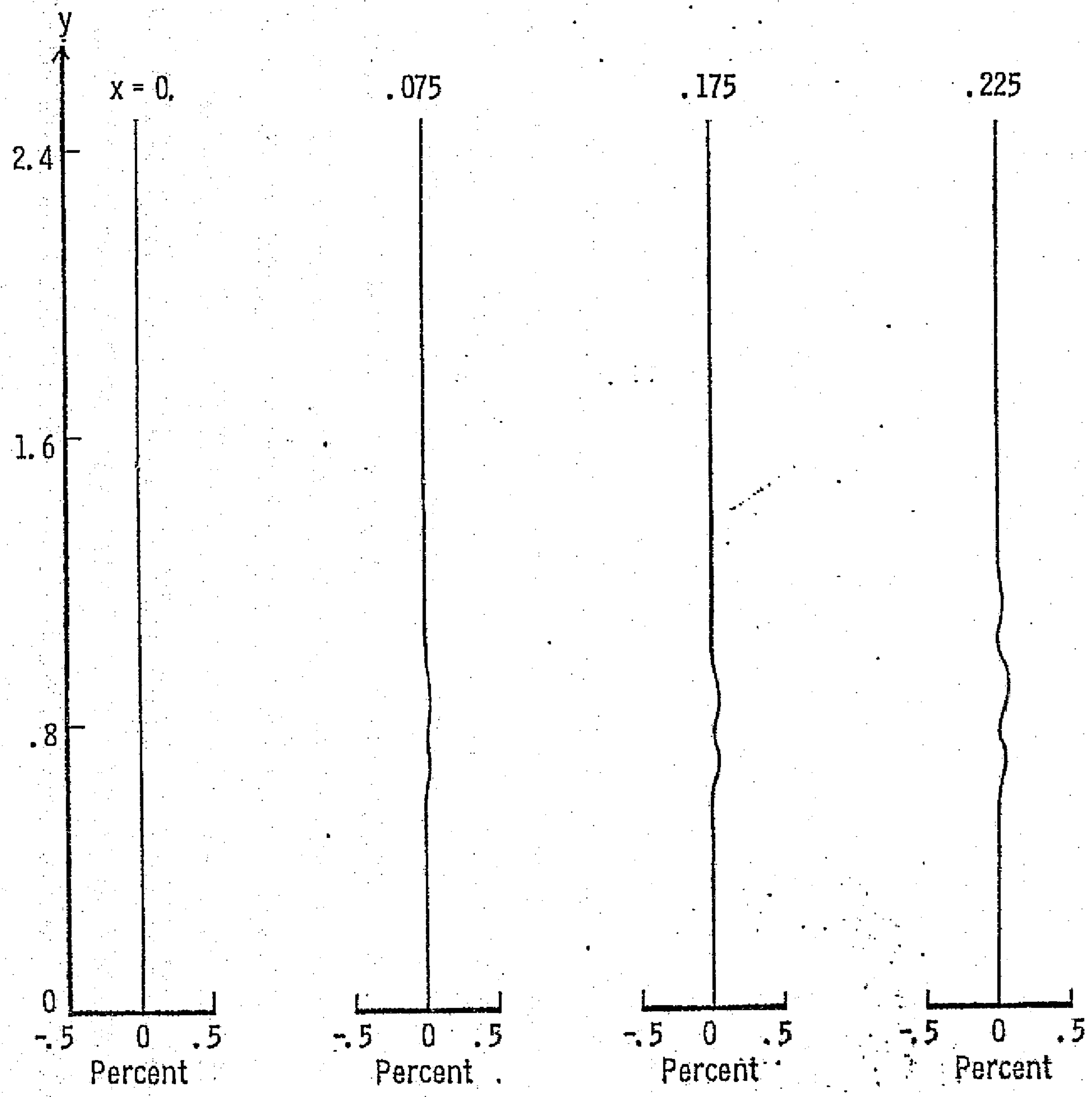


Figure 7. - FEM-ADI percent difference in density.

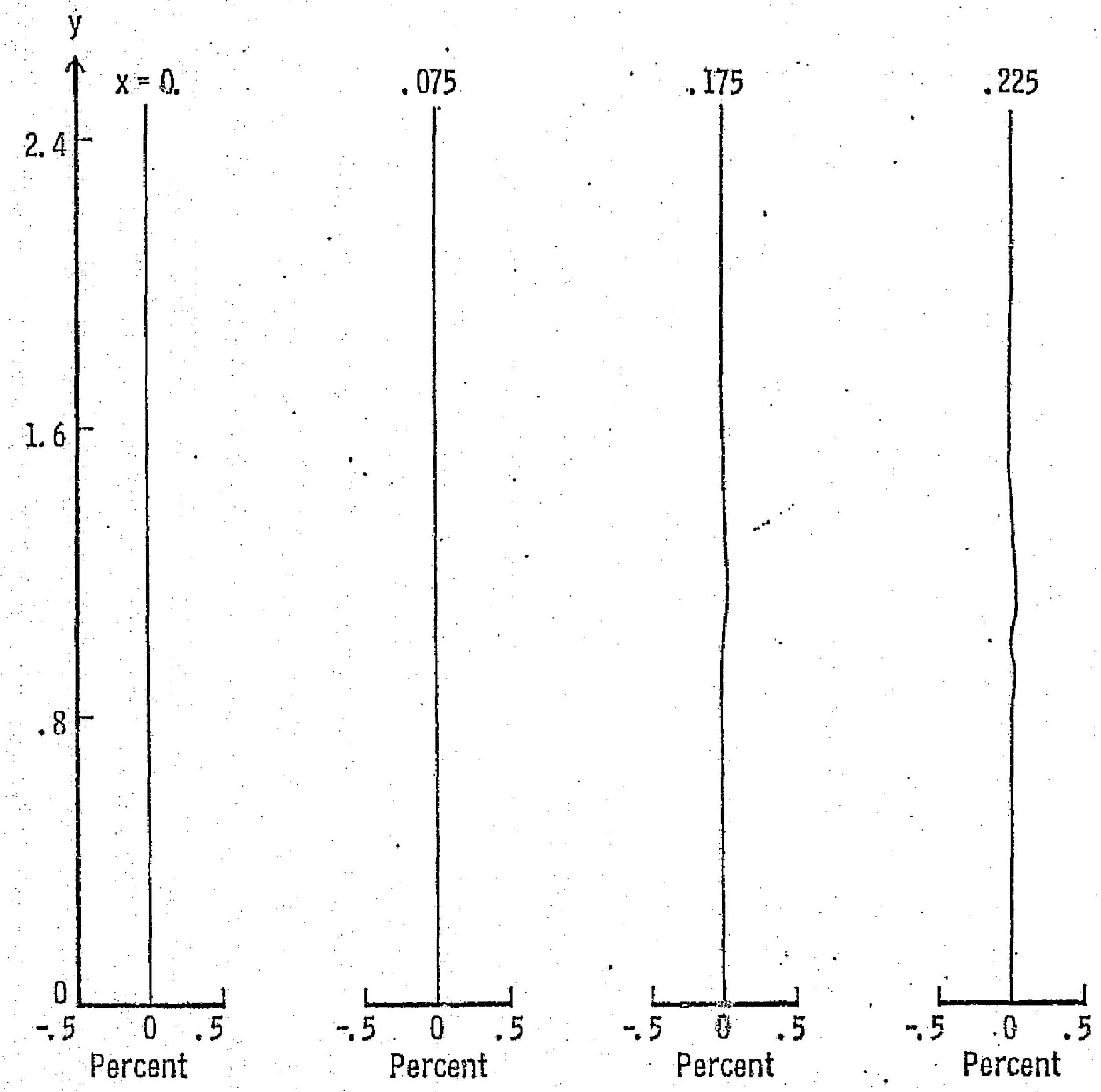


Figure 8. - FEM-ADI percent difference in streamwise velocity component.

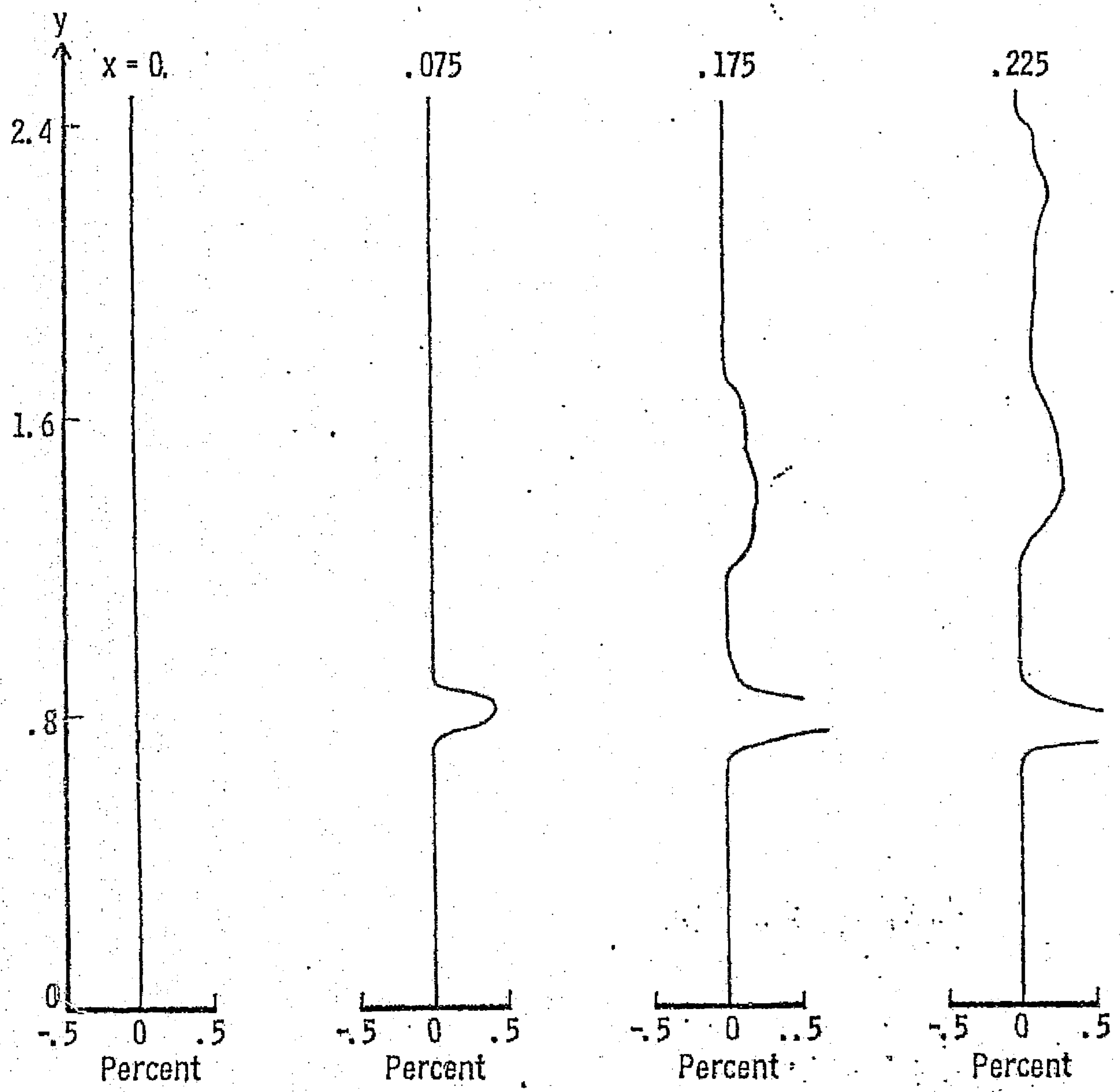
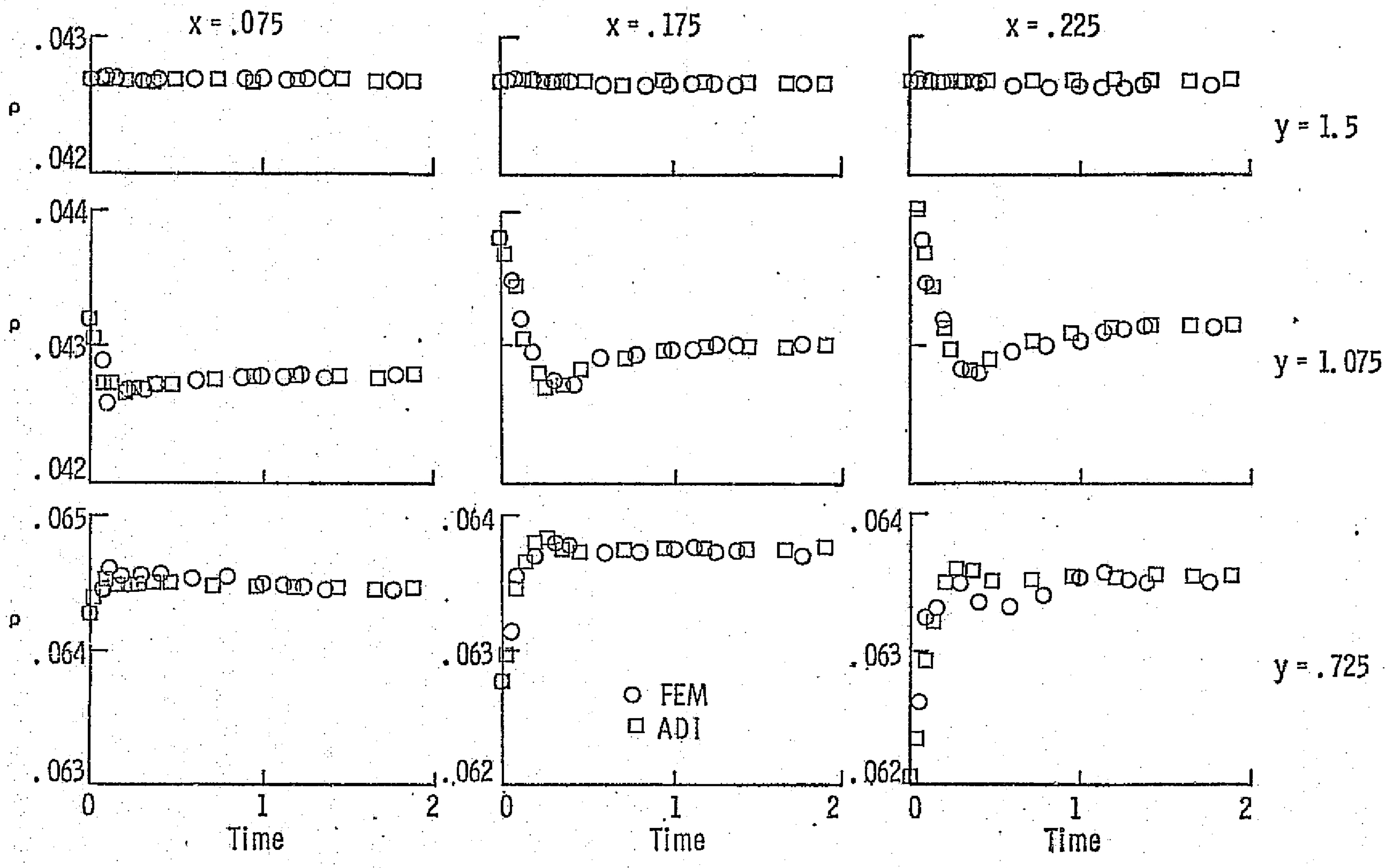
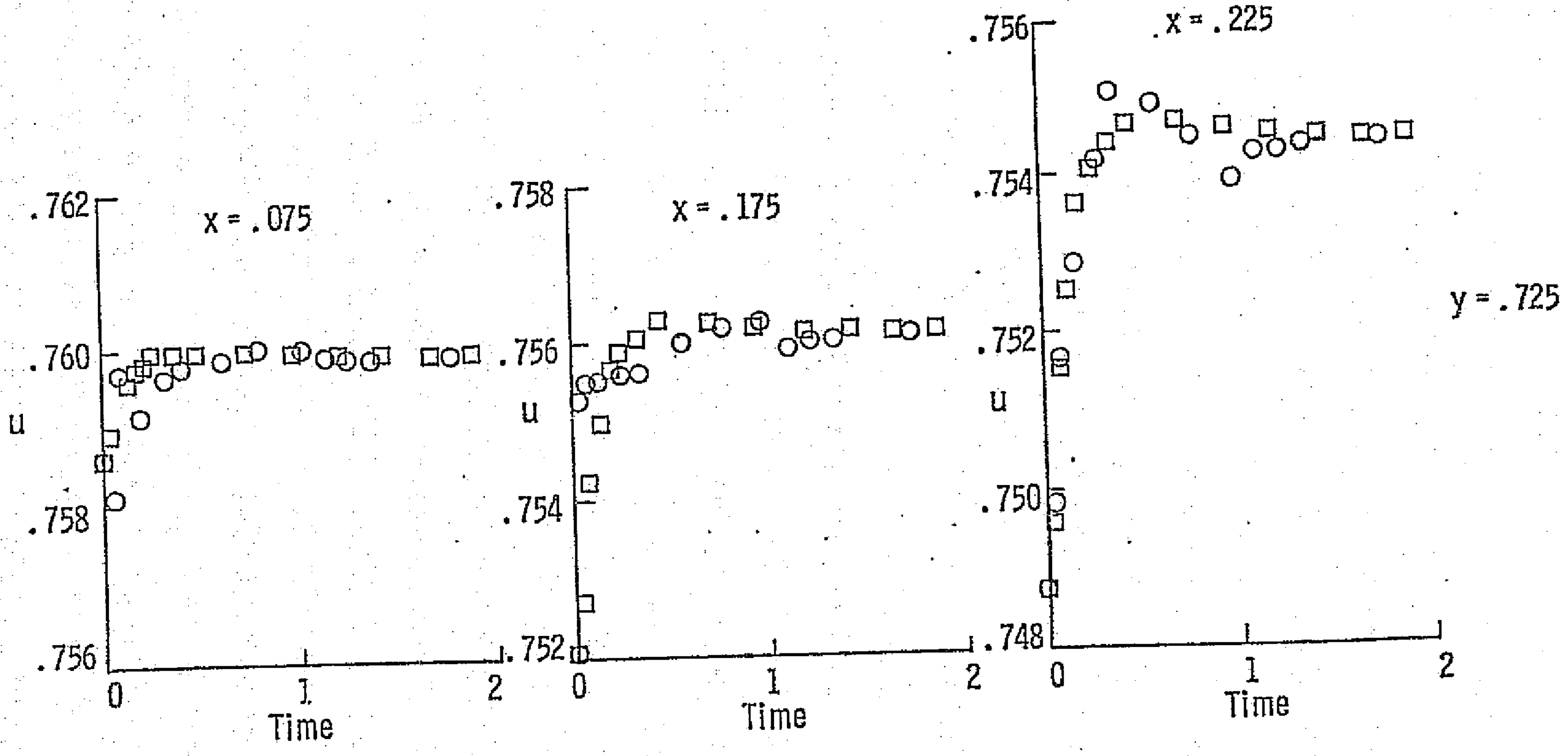


Figure 9. - FEM-ADI percent difference in normal velocity component.



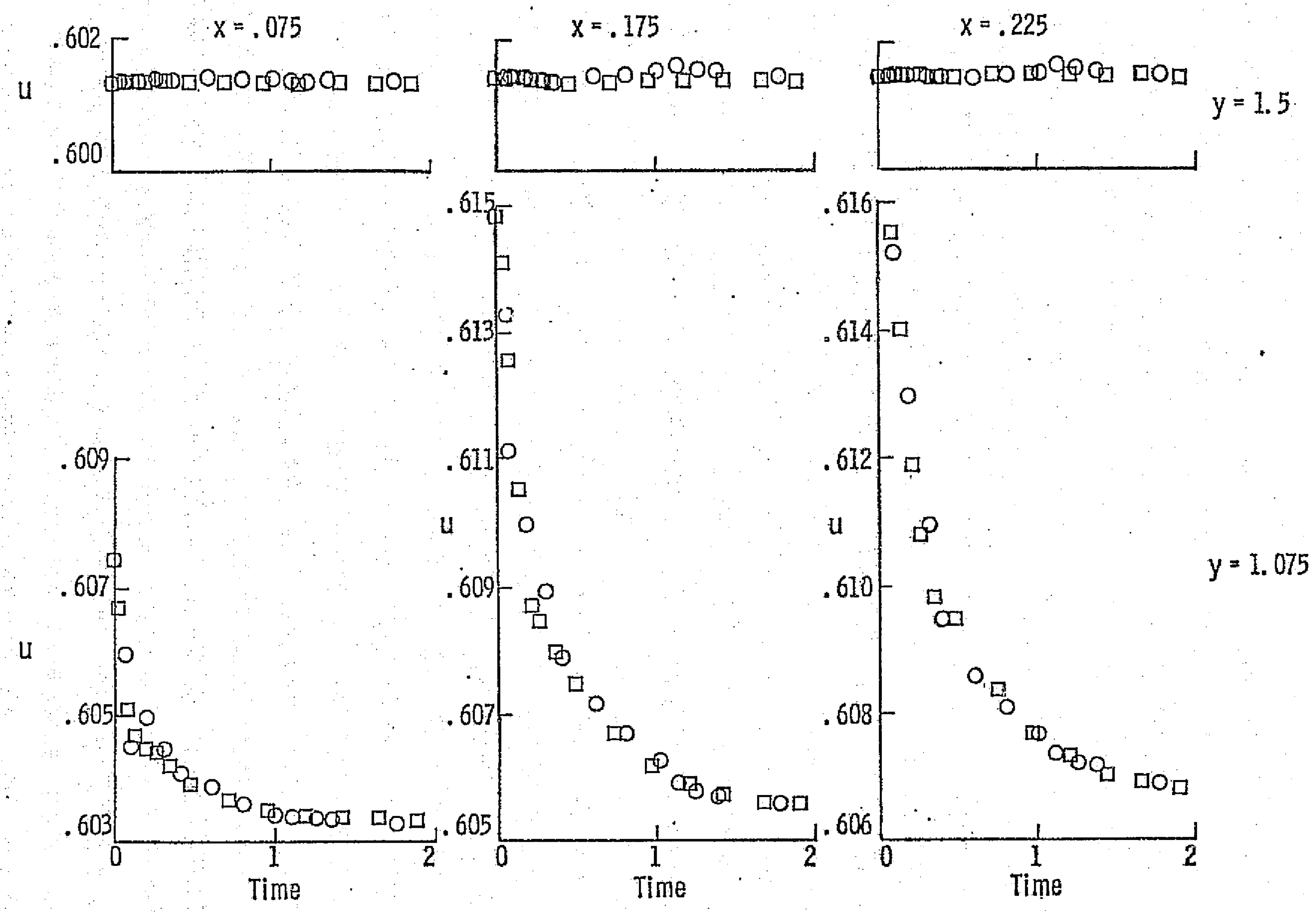
(a) Density.

Figure 10. - FEM-ADI convergence comparisons.



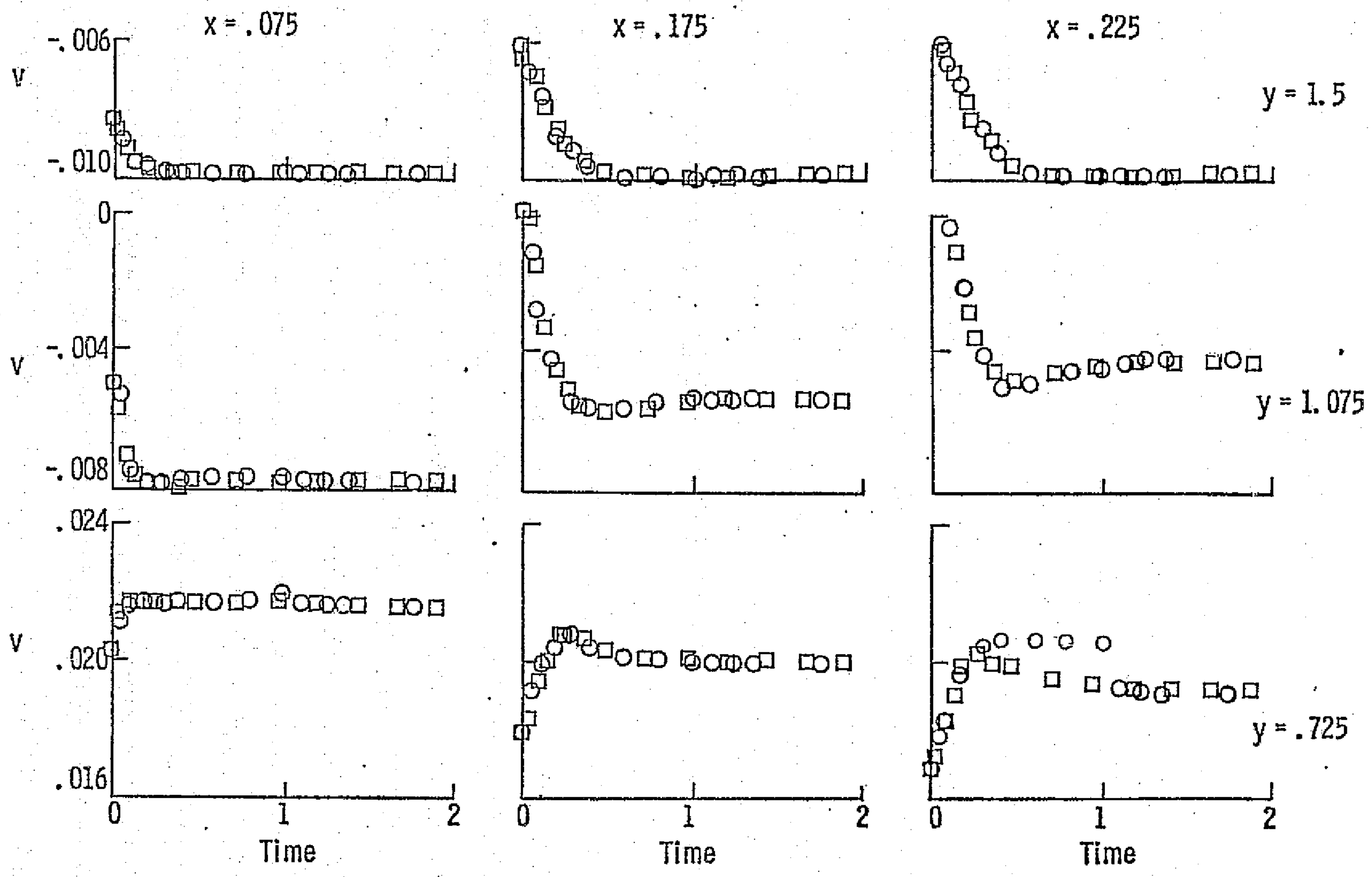
(b) Streamwise velocity.

Figure 10.- Continued.



(b) Concluded.

Figure 10. - Continued.



(c) Normal velocity.

Figure 10. - Concluded.